Analiza mape jakosti elektromagnetskog radiofrekvencijskog polja na temelju skale boja

Đurašević, Mirko

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

Analiza mape jakosti elektromagnetskog radio frekvencijskog polja na temelju skale boja

Završni rad

Mirko Đurašević

Osijek, 2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P: Obrazac za ocjenu završnog rada na sveučilišnom prijediplomskom studiju

Ocjena završnog rada na sveučilišnom prijediplomskom studiju

Ime i prezime pristupnika:	Mirko Đurašević
Studij, smjer:	Sveučilišni prijediplomski studij Elektrotehnika i informacijska tehnologija
Mat. br. pristupnika, god.	4815, 28.07.2020.
JMBAG:	0165087239
Mentor:	prof. dr. sc. Snježana Rimac-Drlje
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Analiza mape jakosti elektromagnetskog radiofrekvencijskog polja na temelju skale boja
Znanstvena grana završnog rada:	**Telekomunikacije i informatika (zn. polje elektrotehnika)**
Zadatak završnog rada:	U proračunima jakosti elektromagnetskog polja u nekom prostoru često se rezultati vizualiziraju skalom boja kojima su pridjeljeni određeni rasponi vrijednosti računate veličine. U radu je potrebno osmisliti algoritam i napraviti program koji će na temelju boja na slici koja predstavlja rezultate proračuna jakosti električnog polja u nekom prostoru izračunati statističke parametre kao što su prosječna jakost električnog polja te postotak površine na kojoj je električno polje veće ili manje od definirane vrijednosti.

Datum prijedloga ocjene završnog rada od strane mentora:	19.09.2024.
Prijedlog ocjene završnog rada od strane mentora:	Vrlo dobar (4)
Datum potvrde ocjene završnog rada od strane Odbora:	25.09.2024.
Ocjena završnog rada nakon obrane:	Vrlo dobar (4)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio sveučilišni prijediplomski studij:	29.09.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 29.09.2024.

Ime i prezime Pristupnika:	Mirko Đurašević
Studij:	Sveučilišni prijediplomski studij Elektrotehnika i informacijska tehnologija
Mat. br. Pristupnika, godina upisa:	4815, 28.07.2020.
Turnitin podudaranje [%]:	11

Ovom izjavom izjavljujem da je rad pod nazivom**: Analiza mape jakosti elektromagnetskog radiofrekvencijskog polja na temelju skale boja**

izrađen pod vodstvom mentora prof. dr. sc. Snježana Rimac-Drlje

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

Sadržaj

1.UVOD

U današnjem društvu elektromagnetska polja imaju ključnu ulogu u modernoj tehnologiji. Primjenjuje se u komunikacijskim sustavima, radarskoj tehnologiji, medicinskoj opremi i mnogim drugima granama tehnologije. Razumijevanje elektromagnetskog polja omogućuje precizno planiranje i optimizaciju tehničkih sustava. Za proračun jakosti električnog i magnetskog polja zračećih struktura razvijeni su razni programi koji koriste numeričke metode za proračun, a metode vizualizacije podataka za prikaz izračunatih fizikalnih veličina. Vizualizacija jakosti električnog/magnetskog polja temelji se na skali boja, koja pruža intuitivnu metodu za razumijevanje složenih podataka. Iako se dobivene vrijednosti jakosti polja mogu dalje direktno analizirati iz numeričkih podataka, u ovom je radu istražena mogućnost analize ovih vrijednosti predstavljenih na slici preko skale boja. Rješenje ovog zadatka temelji se na mapiranju jakosti elektromagnetskog polja pomoću dane skale boja, olakšavajući korisniku unos podataka u program. Korisnik može odabrati vertikalnu liniju na slici koja predstavlja skalu boja. Program zatim mapira boje na odgovarajuće vrijednosti jakosti polja, koristeći RGB prostor boja. Program izračunava ključne statističke parametre za odabranu regiju, poput prosječne jakosti električnog polja, medijana, standardne devijacije, minimalne i maksimalne vrijednosti odabrane regije. Rezultate analize prikazuju se pomoću originalne odabrane regije, 3D prikaza distribucije jakosti polja, histograma i kumulativne distribucijske funkcije. Svrha ovog rada je pružiti inženjerima alat koji omogućava brzu i učinkovitu analizu jakosti električnog/magnetskog polja pomoću vizualnih podataka u slučaju kada je dostupna samo slika proračunatih vrijednosti, a ne i numerički podaci.

1.1 Zadatak završnog rada

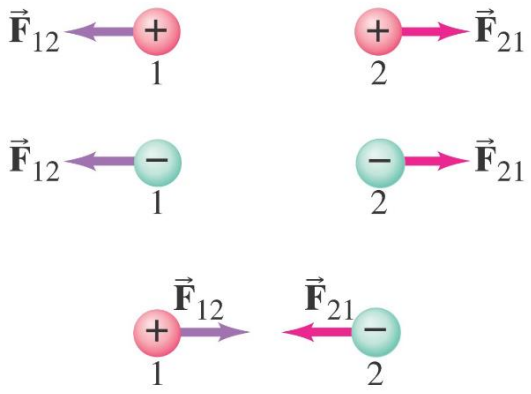
U proračunima jakosti elektromagnetskog polja u nekom prostoru često se rezultati vizualiziraju skalom boja kojima su pridijeljeni određeni rasponi vrijednosti računate veličine. U radu je potrebno osmisliti algoritam i napraviti program koji će na temelju boja na slici koja predstavlja rezultate proračuna jakosti električnog polja u nekom prostoru izračunati statističke parametre kao što su prosječna jakost električnog polja te postotak površine na kojoj je električno polje veće ili manje od definirane vrijednosti.

2. TEORIJSKA POZADINA

2.1. Električni naboj, sila i polje

Električni naboj je fizička veličina koja opisuje elektromagnetske interakcije između čestica. Električni naboj može biti negativan ili pozitivan, mjeri se u kulonima (C), jedinica naboja je elementarni naboj (e). Proton je subatomska čestica koja nosi pozitivan elementarni naboj (+e), elektron je subatomska čestica koja nosi negativan elementarni naboj(-e), gdje je e približno jednako 1.602×10^{-19} kulona[1]. Svojstva električnog naboja su[2]:

- Privlačenje i odbijanje: Čestice s istim tipom naboja se odbijaju, dok se čestice sa suprotnim tipovima naboja privlače, prema slici 2.1
- Kvantizacija: Električni naboj je kvantiziran, što znači da dolazi u diskretnim jedinicama koje su višekratnici elementarne jedinice naboja (e). Svaki naboj u prirodi mora biti $n * e$ gdje je n cijeli broj
- Aditivnost: Električni naboj je aditivan, ukupni naboj sustava može se dobiti zbrajanjem pojedinačnih naboja unutar sustava.
- Očuvanje naboja: Očuvanje naboja govori kako naboj ne može biti stvoren ili uništen. Naboj možemo prebaciti iz jednog oblika u drugi. U izoliranom sustavu naboj je uvijek očuvan.



Sl. 2.1 Djelovanje sile između nabijenih čestice,[3]

Električna sila je fizikalna veličina koja djeluje između električno nabijenih tijela. Ona je rezultat međudjelovanja električnih naboja i manifestira se kao privlačna ili odbojna sila između čestica. Predstavlja jednu od četiri osnovne sile u prirodi, koje dodatno uključuju gravitaciju , slabu i jaku nuklearnu silu[1]. Može se opisati sljedećom formulom :

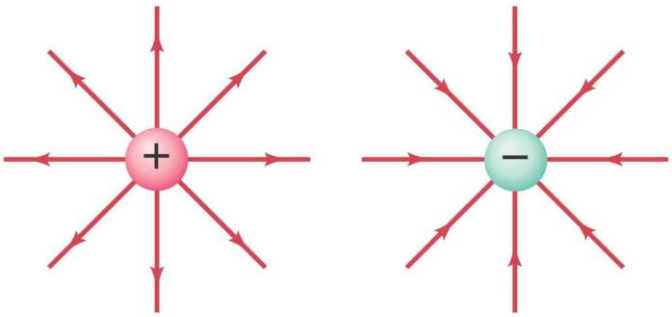
$$F = k * \frac{Q_{1*}Q_2}{r^2} \qquad (2-1)$$

Gdje F predstavlja silu između naboja, k je Coulombova konstanta, Q_1 i Q_2 predstavljaju veličinu naboja, r predstavlja udaljenost između njih. U vakuumu Coulombova konstanta iznosi $8.99 * 10^9\ Nm^2/C^2$ [1]:

Električno polje je fizikalna veličina koja opisuje vektorsko polje koje okružuje električni naboj. Električno polje definira se kao silu koja djeluje na jedinicu naboja koju bi čestica osjetila kada bi se nalazila u tom prostoru. Definirana je formulom :

$$\vec{E} = \frac{\vec{F}}{q} \qquad (2-2)$$

gdje je F sila koja djeluje na jedinicu naboja, a q veličina naboja. Mjerna jedinica je njutn po kulonu ($\frac{N}{C}$), odnosno volt po metru (V/m). Silnice električnog polja predstavljaju zamišljene linije koje se koriste za vizualizaciju električnih polja. Linije predstavljaju smjer i jakost električnog polja u prostoru oko naelektriziranih čestica, prikazano slikom 2.2 [4].



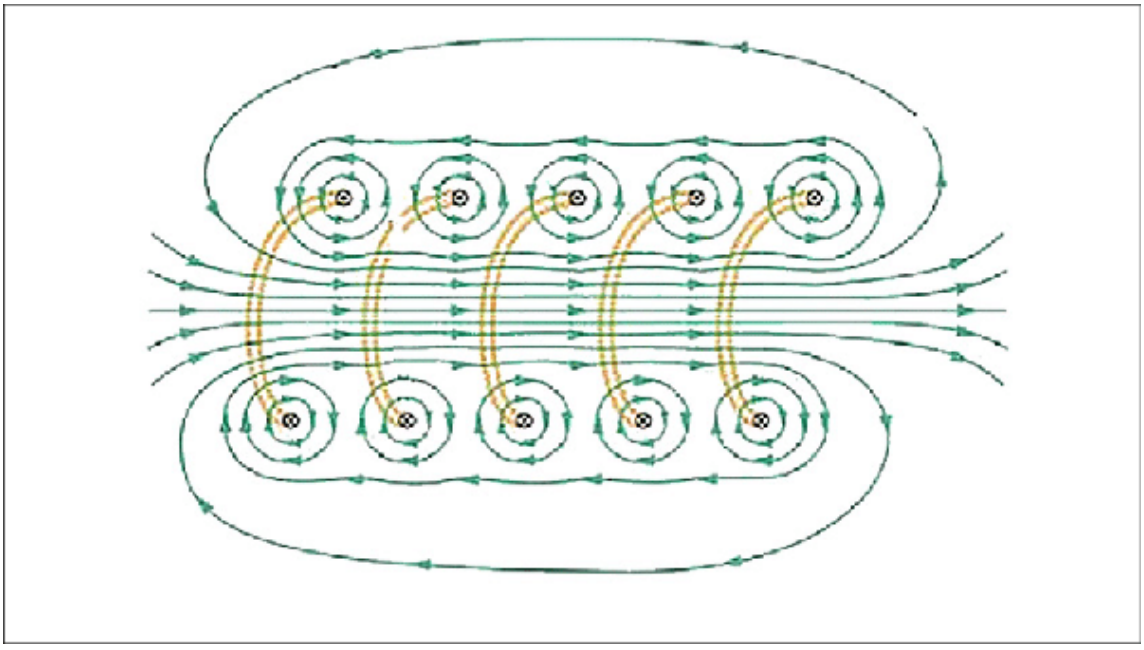
Sl. 2.2 Prikaz linija električnog polja pozitivnog i negativnog naboja,[3]

Svojstva električnog polja su:

- Superpozicija: Kada je prisutno više električnih polja, ukupno vektorsko polje je vektorski zbroj pojedinačnih polja.
- Vektor: Električno polje je vektorska veličina, što znači da ima magnitudu i smjer. Smjer vektora električnog polja definiran je kao smjer sile koja djeluje na pozitivnu naelektriziranu česticu postavljenu u tom polju.
- Očuvanje električnog polja: Govori kako rad obavljen pri premještanju električnog naboja između dvije točke ne ovisi o putu, nego samo o početnoj i krajnjoj točki

2.2. Magnetsko polje

Magnetsko polje predstavlja vektorsko polje koje opisuje djelovanje magnetske sile na magnetske dipole i pokretne električne naboje. Za razliku od električnog naboja, za magnetski naboj još uvijek nije potvrđeno da postoji. Kada se električni naboj kreće on stvara magnetsko polje oko sebe. Primjer toga je struja koja teče kroz žicu. Ako se žica namota u spiralu i kroz nju prođe električna struja, stvara se magnetsko polje slično onom kod štapnog magneta prikazano na slici 2.3. Jakost magnetskog polja u tom slučaju ovisi o jačini struje i broj zavoja.



Sl. 2.3 Prikaz toka magnetskog polja kroz zavojnicu,[5]

Magnetsko polje također može nastati kod materijala koji imaju trajna magnetska svojstva. Oni imaju stabilno magnetsko polje bez potrebe za vanjskim izvorom energije. Primjeri takvih materijala uključuju željezo, kobalt i nikl. Za opis sile koja djeluje na naelektriziranu česticu u magnetskom polju koristi se formula za Lorentzovu silu:

$$\vec{F}_B = q(\vec{v} \times \vec{B}) \qquad (2-3)$$

gdje je F_B sila magnetskog polja koja djeluje na česticu, q električni naboj čestice, a $\vec{v} \times \vec{B}$ vektorski umnožak brzine čestice i vektora magnetskog polja.

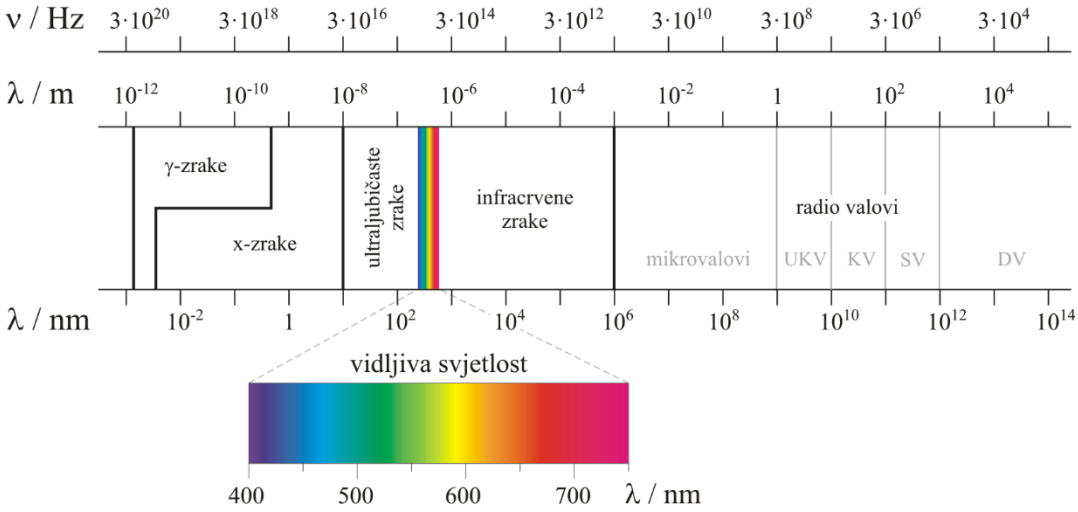
2.3. Elektromagnetski valovi

Elektromagnetski valovi su valovi koji ne koriste medij za prijenos energije, mogu se kretati kroz vakuum. Primjeri uključuju svjetlost, radio valove, x-zrake i televizijske valove. U vakuumu oni putuju brzinom svjetlosti. Sastoje se od oscilirajućih električnih i magnetskih polja koja osciliraju okomito jedno na drugo i na smjer širenja vala.

Najvažnije karakteristike elektromagnetskih valova uključuju:

- Valna duljina (λ): Udaljenost između dva uzastopna vrhunca ili doline vala
- Frekvencija(f):Broj valnih ciklusa koji prođu kroz određenu točku u jednoj sekundi
- Brzina svjetlosti(c): Brzina kojom se elektromagnetski valovi šire kroz vakuum
- Polarizacija: Smjer oscilacije polja u valovima

Elektromagnetski valovi imaju različite frekvencije. Skup svih mogućih frekvencija čine elektromagnetski spektar. Unutar tog spektra, postoji dio vidljiv ljudskom oku, poznat kao vidljiva svjetlost. Vidljiva svjetlost obuhvaća mali dio ukupnog elektromagnetskog spektra, u kojem su i radio valovi, mikrovalovi, infracrvene zrake, ultraljubičaste zrake, x-zrake, i gama zrake kao što je prikazano na slici 2.4 [6].



Sl. 2.4 Spektar elektromagnetskih valova s istaknutim dijelom vidljive svijetlosti,[7]

Elektromagnetski val koji putuje po x osi , sastoji se od električnog polja i magnetskog polja čije magnitude ovise o poziciji x osi i vremenu. Njihovo ponašanje možemo opisati sljedećim formulama:

$$E = E_m \sin(kx - \omega t) \qquad (2-4)$$

$$B = B_m \sin(kx - \omega t) \qquad (2-5)$$

E_m i B_m predstavljaju amplitude električnog i magnetskog polja, ω predstavlja kružnu frekvenciju koja iznosi $2f\pi$. Električno polje inducira magnetsko polje i obratno.

2.4. Fizikalni principi elektromagentskog polja

Elektromagnetsko polje sastoji se od međusobno povezanog električnog i magnetskog polja. Električno polje nastaje oko električnog naboja, dok magnetsko polje nastaje gibanjem tih naboja. Osnovni fizikalni zakoni koji opisuju pojave u eklektičnom i magnetskom polju su Gaussov, Faradayev i Amperov zakon. Na temelju ovih zakona Maxwellov je razvio jednadžbe koje opisuju elektromagnetska polja i valove. Maxwellove jednadžbe objašnjavaju potrebnu teorijsku osnovu za razumijevanje ponašanje elektromagnetskih polja i kako se elektromagnetska sila prenosi kroz prostor i vrijeme, [8].

- Gaussov zakon za električno polje kaže da je ukupni električni tok kroz bilo koju zatvorenu površinu proporcionalan ukupnom električnom naboju sadržanom unutar te površine. Matematički, Gaussov zakon se može izraziti kao

$$\oint \vec{E} \cdot \mathrm{d}\vec{A} = \frac{Q_{uk}}{\varepsilon_0} \qquad (2-6)$$

$\vec{E}$ predstavlja vektor električnog polja, $\mathrm{d}\vec{A}$ vektor površine, Q_{uk} ukupni naboj unutar zatvorene površine i ε_0 je permitivnost vakuuma. $\oint$ označava integral po zatvorenoj površini. Gaussov zakon povezuje izvor električnog polja – naboje i samo polje.

- Gaussov zakon za magnetsko polje ili zakon o konzervaciji magnetskog polja kaže da je ukupni magnetski tok kroz zatvorenu površinu uvijek jednak nula, stoga za razliku od električnog naboja, magnetski monopoli ne postoje u prirodi. Matematički, Gaussov zakon za magnetsko polje može se izraziti kao

$$\oint \vec{B} \cdot d\vec{A} = 0 \qquad (2-7)$$

$\vec{B}$ predstavlja vektor magnetske indukcije. Magnetsko polje se uvijek pojavljuje u obliku zatvorenih silnica bez početka ili kraja. Zbog nepostojanja magnetskog monopola, rad potreban za pomicanje magnetskog pola po zatvorenoj putanji uvijek je nula.

- Faradayev zakon elektromagnetske indukcije govori kako promjena magnetskog polja unutar zatvorene petlje inducira električni napon u petlji. Koncept je vezan uz rad transformatora, gdje se mehanička energija pretvara u električnu koristeći promjene u magnetskom polju. Matematički, Faradayev zakon se može izraziti kao

$$\varepsilon = -N * \frac{d\Phi}{dt} \qquad (2-8)$$

- ε predstavlja induciranu elektromotornu silu, N broj namotaja u zavojnici, $d\Phi/dt$ brzina promjene magnetskog toka. Elektromotorna sila može se inducirati na sljedeće načine: promjenom jakosti magnetskog polja, promjenom površine u magnetskom polju,

promjenom orijentacije petlje u odnosu na magnetsko polje. S obzirom da napon postoji samo ako u prostoru postoji električno polje, ovaj zakon se može interpretirati i kao veza između promjenjivog magnetskog polja i električnog polja koje takvo magnetsko polje stvara. To je jedan od ključnih elementa u generiranju elektromagnetskog vala, a može se iskazati sljedećim izrazom:

$$\oint \vec{E} \cdot d\vec{l} = -\frac{d}{dt}\iint \vec{B}d\vec{S} \qquad (2-9)$$

- Ampere-Maxwellov zakon proširuje Amperov zakon uključivanjem promjenjivog električnog polja kao dodatnog izvora magnetskih polja. Ovaj zakon povezuje električnu struju i magnetsko polje koje ta struja stvara. Matematički, Ampère-Maxwellov zakon se može izraziti kao

$$\oint \vec{B} \cdot d\vec{l} = \mu_0 I \qquad (2-10)$$

Ampère-Maxwellov zakon, zajedno s Faradayevim zakonom, omogućuje matematički opis propagacije elektromagnetskih valova kroz prostor.

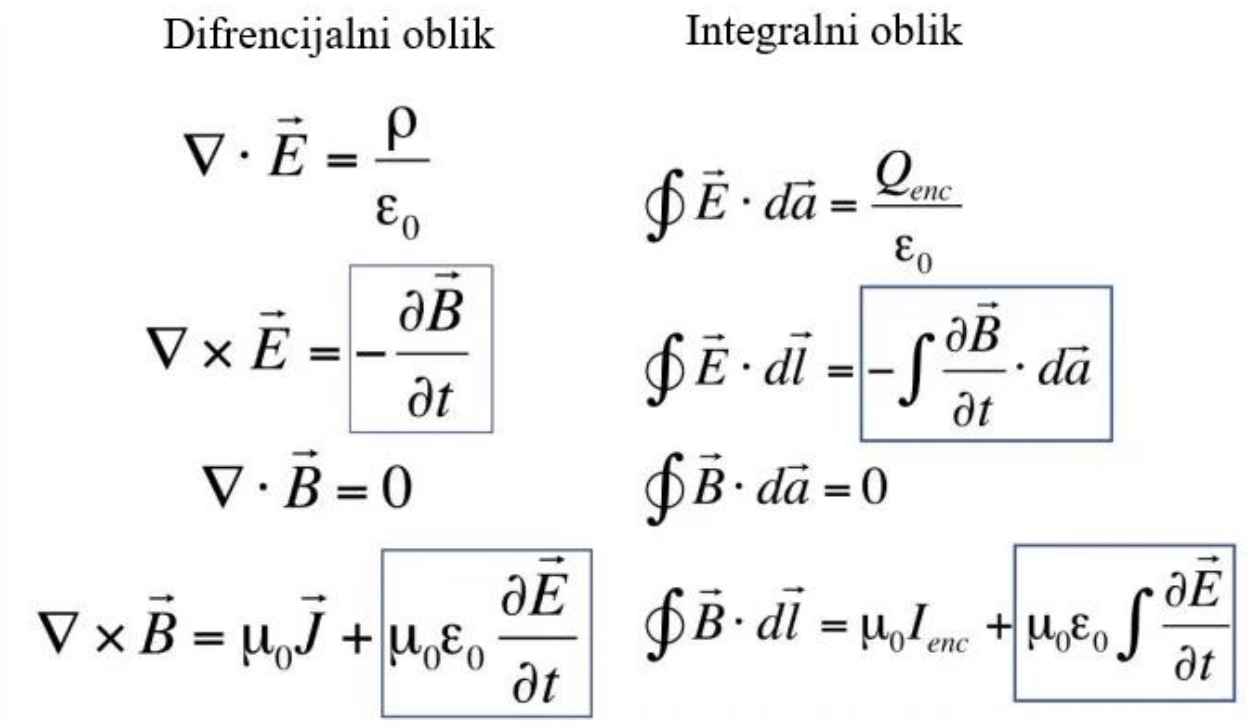
- Lorentzova sila opisuje silu koja djeluje na nabijenu česticu kada se kreće kroz magnetsko polje. Predstavlja temeljni mehanizam za razumijevanje kako elektroni reagiraju unutar elektromagnetskog polja, omogućavajući razumijevanje pojmova poput magnetskog polja i električne struje. Lorentzova sila se može izraziti kao

$$\vec{F} = q\left(\vec{E} + \vec{v} \times \vec{B}\right) \qquad (2-11)$$

Prvi dio jednadžbe ($q\vec{E}$) predstavlja silu zbog električnog polja, koja djeluje u smjeru električnog polja, ovisno o predznaku naboja. Drugi dio $q(\vec{v} \times \vec{B})$ predstavlja silu zbog magnetskog polja, koja je uvijek okomita na smjer kretanja čestice i smjer magnetskog polja..

Maxwellove jednadžbe i Lorentzova sila pružaju razumijevanje elektromagnetskih fenomena, što omogućuje pravljenje tehnoloških aplikacija koje uključuju elektromagnetsku interakciju,

poput radiokomunikacija i medicinskih uređaja. Maxwelove jednadžbe prikazane sljedećim izrazima



Sl. 2.5 Prikaz Maxwellovih jednadžbi u diferencijalnom i integralnom obliku,[9]

Analitičko rješenje Maxwellovih jednadžbi moguće je samo za jednostavne zračeće strukture, pa se za proračune elektromagnetskog polja za većinu realnih izvora zračenja koriste numeričke metode. Najčešće korištene numeričke metode uključuju:

- Metoda momenata (eng. *Method of Moments* - MoM): Ova metoda je posebno učinkovita za probleme koji uključuju žičane antene i metalne površine. MoM se temelji na pretvorbi integralnih jednadžbi u sustav linearnih algebarskih jednadžbi. Učinkovita za probleme zračenja i raspršenja, ali računski zahtjevna za velike strukture.
- Metoda konačnih elemenata (eng. *Finite Element Method* -FeM): koristi se za analizu složenih geometrija i nehomogenih materijala. FEM dijeli promatrano područje na manje elemente (najčešće trokute ili tetraedre) i rješava parcijalne diferencijalne jednadžbe za svaki element. Vrlo prilagodljiva, pogodna za složene geometrije i nehomogene materijale, ali ujedno i računski zahtjevna, posebno za 3D probleme.
- Metoda konačnih razlika u vremenskoj domeni (eng. *Finite-Difference Time-Domain* - FDTD): FDTD je popularna metoda za rješavanje problema u vremenskoj domeni.

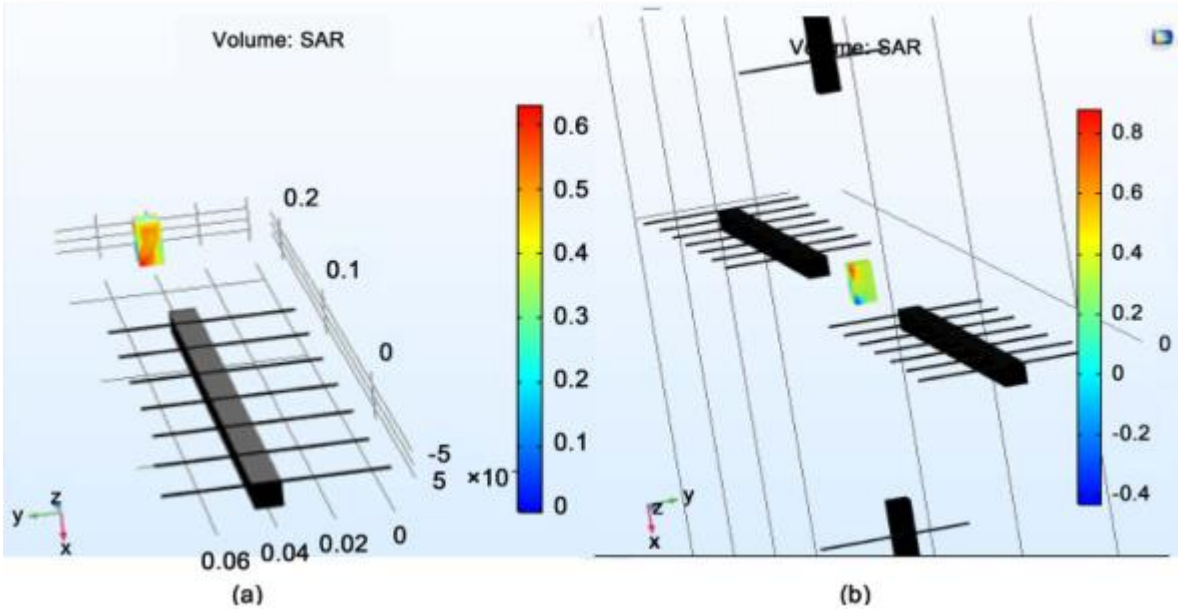
FDTD diskretizira prostor i vrijeme, rješavajući Maxwell-ove jednadžbe direktno u vremenskoj domeni. Pogodna za širokopojasne probleme, jednostavna implementacija, zahtijeva fine mreže za točne rezultate,.

- Metoda konačne integracije (eng. *Finite Integration Technique* -FIT): FIT se temelji na integralnom obliku Maxwell-ovih jednadžbi. Primjenjiva je i u vremenskoj i u frekvencijskoj domeni. Manje učinkovita od specijaliziranih metoda za određene tipove problema.
- Metoda rubnih elemenata (eng. *Boundary Element Method* -BEM): fokusira se na granice domena. Učinkovita za probleme s otvorenim granicama, smanjuje dimenzionalnost problema, ali manje učinkovita za nehomogene materijale, rezultirajući gustim granicama.

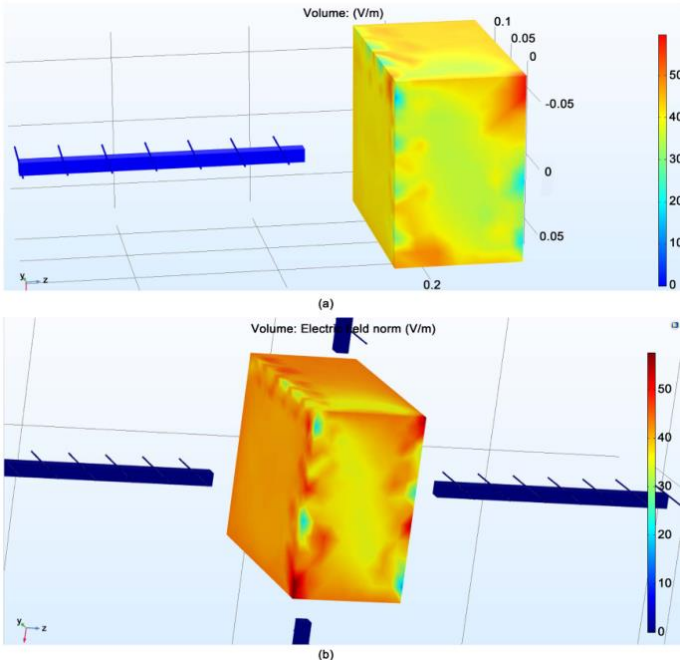
2.5 Proračun veličina elektromagnetskog polja primjenom programskih alata

Precizan proračun elektromagnetskog polja omogućava optimizaciju izvedbi različitih uređaja, primjerice poboljšanje efikasnosti antena, transformatora i elektromotora. Također se osigurava da elektromagnetska polja poštivanju međunarodne standarde i ne prelaze sigurnosne granice koje bi mogle štetiti ljudima. Za ove proračune razvijeni su razni programski alati koji koriste numeričke metode za proračun jakosti električnog i/ili magnetskog polja u okolini zračećih elemenata.

Jedan od najpoznatijih programskih alata za proračun veličina elektromagnetskog polja je ANSYS. Između ostalog, koristi se za dizajn i analizu radio frekvencijskih komponenti poput filtera i antena. Sastoji se od Ansys Mechanical modula fokusiran na analizu naprezanja i vibracija, Ansys Fluent modula za simulaciju mehanike tekućina i Ansys HFSS **(High-Frequency Structure Simulator)** modula za trodimenzionalnu elektromagnetsku simulaciju visokih frekvencija. Ovaj alat je ključan za dizajn antena, analizu elektromagnetske kompatibilnosti i proračun specifične gustoća apsorbirane snage (eng. *Specific Absorption Rate* - SAR) u biološkim tkivima, [10]. Tako autori u [11] koriste ANSYS HFSS i COMSOL za simulaciju SAR vrijednosti u modelu ljudske glave. Analizirani su utjecaji različitih postavki antena na raspodjelu elektromagnetskih polja i SAR, gdje su simulacije pomogle u optimizaciji dizajna antene za biomedicinske svrhe.



Sl. 2.6 Prikaz SAR mjerenja za (a) jednu antenu i (b) četiri antene,[11]

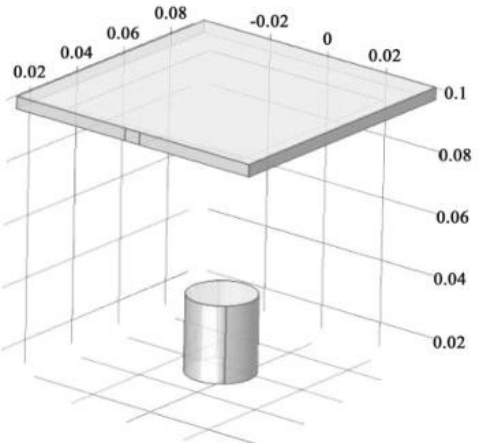


Sl. 2.7 Distribucija električnog polja za (a) jednu antenu i (b) četiri antene,[11]

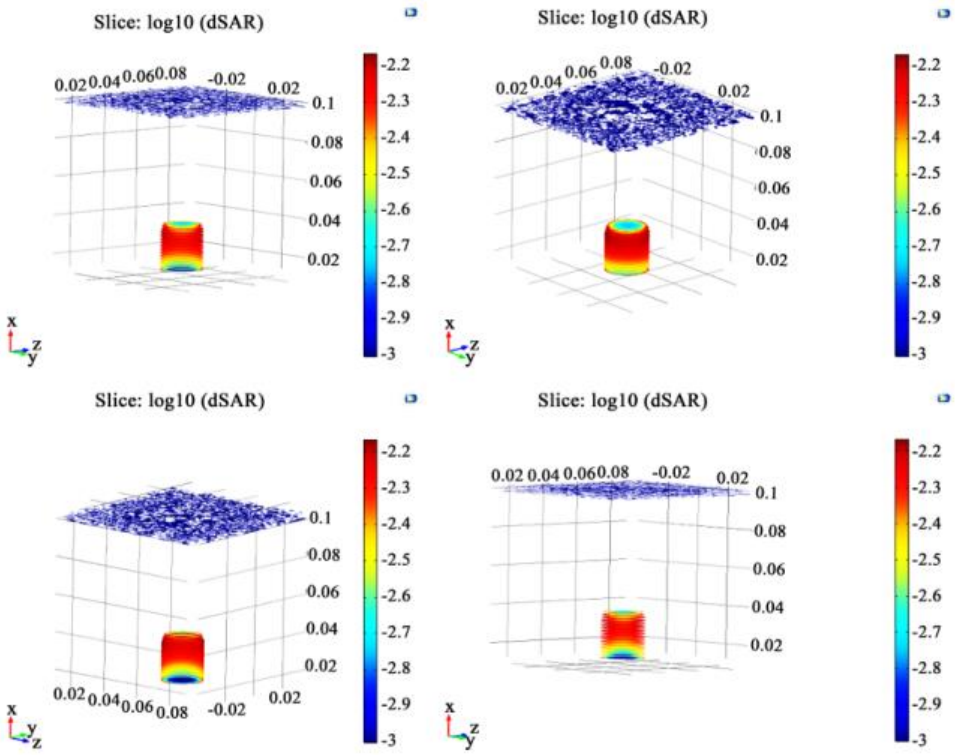
Primjeri primjene uključuju modeliranje antena za medicinske aplikacije, poput dijagnozirana Alzheimerove bolesti i simulacije elektromagnetskog polja unutar ljudske glave.

COMSOL Multiphyscis [12] je softver koji se najčešće koristi za simulaciju interakcije između različitih fizičkih fenomena poput tekućina, toplinskih prijenosa i elektromagneta. Sastoji se od više modula od kojih su glavni AC/DC modul koji se koristi za analizu elektromagnetskih polja, električnih krugova, CFD modul fokusiran na mehaniku tekućina i prijenos topline i Optics

modula korišten za simulaciju optičkih komponenti i svjetlosnih pojava. Primjer korištenja COMSOL softvera možemo vidjeti u [13]. COMSOL je korišten za proračun toplinskog profila i elektromagnetske raspodjele u neuronima. Rezultati su pokazali uniformna elektromagnetska polja s minimalnim toplinskim efektima, slika 2.8 i 2.9.



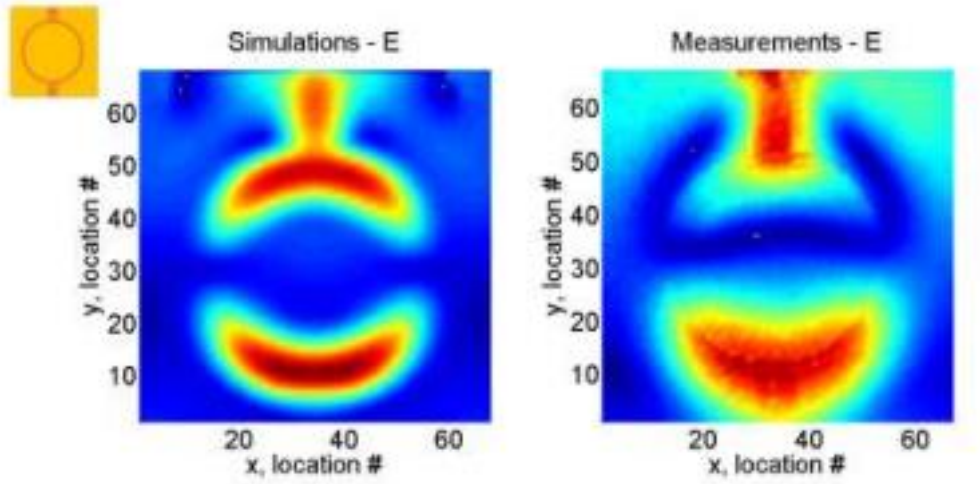
Sl. 2.8 Antena korištena za generiranje elektromagnetskog vala,[13]



Sl. 2.9 Rezultati simulacije korištenjem COMSOL programa,[13]

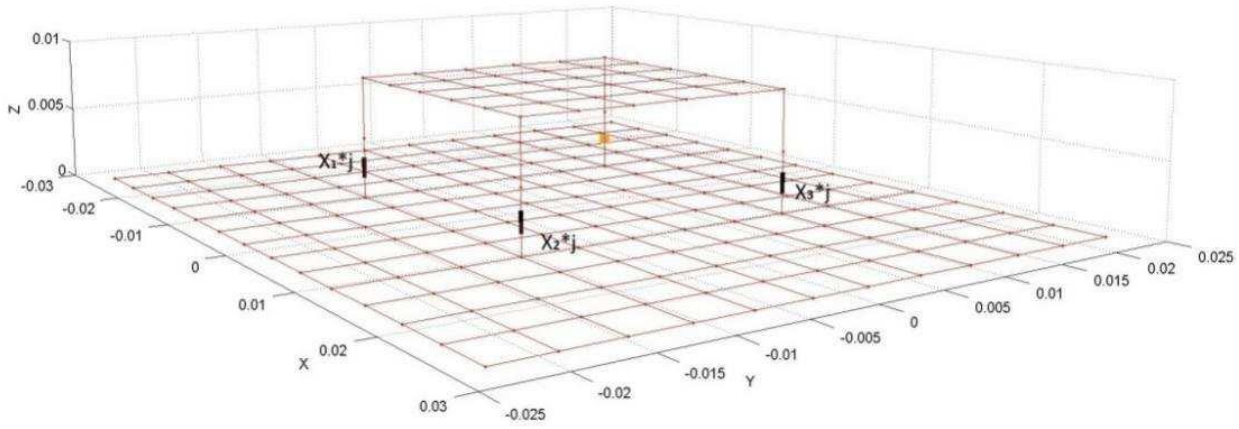
CST Studio Suite [14] je softverski paket korišten za elektromagnetske simulacije. Koristi se u različitim industrijama za analizu elektromagnetskih uređaja. Sastoji se od CST MWS modula koji je specijaliziran za simulacije visokofrekvencijskih elektromagnetskih polja,

CST EM modula fokusiran na simulacije nisko frekvencijskih elektromagnetskih polja, CST PS modula korišten za simulaciju čestica u elektromagnetskim poljima i CST CS modula koje se koristi za analizu elektromagnetske kompatibilnosti i elektromagnetske smetnje u kabelima i žičnim mrežama. CST koristi metodu konačnih integrala za precizne simulacije elektromagnetskih polja. Često se koristi u dizajnu antena i EMC analizi. U [15] se kombinirana mjerenja i simulacije kako bi se analizirali efekti nano materijala na distribuciju elektromagnetskih polja u blizini elektroničkih sklopova. Za usporedbu simularnih i izmjerenih vrijednosti jakosti električnog polja korišteni su HFSS i CST, slika 2.10.



Sl. 2.10 Vizualizacija jakosti električnog polja korištenjem CST (lijevo) i mjerenja (desno), [15]

Super NEC [16] je alat za simulaciju i optimizaciju antena, koji koristi pristup temeljen na metodi momenta. Optimizacija performansi antene često se provodi korištenjem genetskih algoritama, što omogućuje postizanje boljih struktura antena za širok spektar frekvencijskih opsega, .U [17] korištenjem Super NEC i HFSS programa napravljen je dizajn i optimizacija antene (slika 2.13.) koja može raditi na različitim frekvencijama, što je često korisno u biomedicinskim mjerenjima.



Sl. 2.13 Prikaz antene korištenjem Super NEC platforme, [17]

3.VIZUALIZACIJA PODATAKA

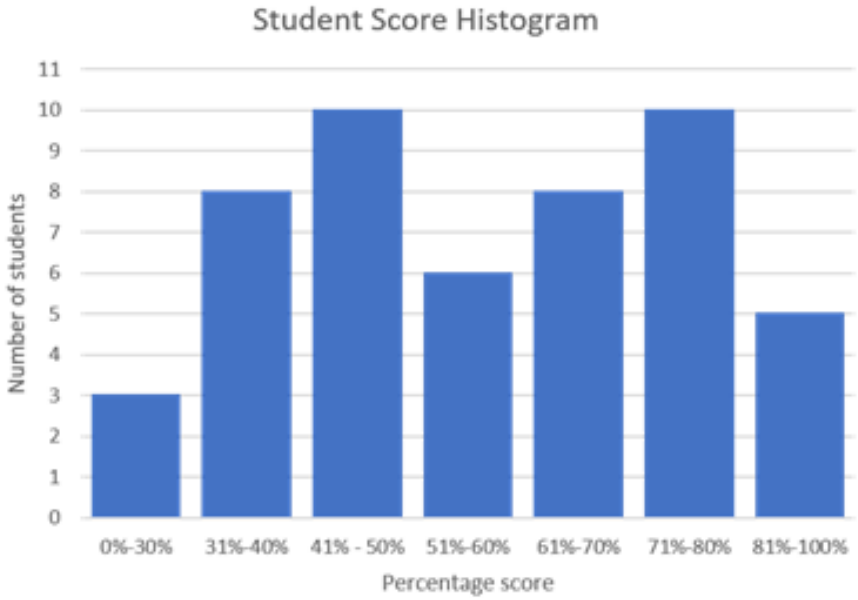
3.1. Vizualizacija rezultata

Vizualizacija igra ključnu ulogu u preoblikovanju kompleksnih podataka u razumljive i korisne informacije. Proces vizualizacije omogućuje korisniku dublje razumijevanje prikupljenih podatka. Vizualizacija elektromagnetskih polja olakšava njihovu analizu i interpretaciju. Za prikaz elektromagnetskih polja posebno su se prikazale korisne sljedeće tehnike:

- Toplinske karte(mape): Koriste se za prikazivanje distribucije elektromagnetskog polja. Ove mape koriste boje za označavanje različitih intenziteta. Intenzivnije boje, poput crvene ukazuju na visoke vrijednosti, dok hladnije boje poput plave ukazuju na niže vrijednosti. Toplinske karte široko se koriste u telekomunikacijama za kartiranje pokrivenosti signalom te identifikaciju područja slabog signala. Toplinske karte su široko korištene tehnike za vizualizaciju prostorne raspodjele elektromagnetskih veličina poput jakosti polja, gustoće snage i stope apsorpcije. Primjeri toplinskih karti u analizi elektromagnetskih polja uključuju vizualizaciju raspodjele gustoće magnetskog toka, mapiranje jakosti električnog polja antena i procjenu apsorpcije raspodjele u biološkim tkivima izloženih radio frekvencijskom zračenju. Toplinske karte omogućuju brzu identifikaciju područja interesa, međutim one su manje učinkovita za prikaz finih detalja i kvantitativnih informacija,[19].

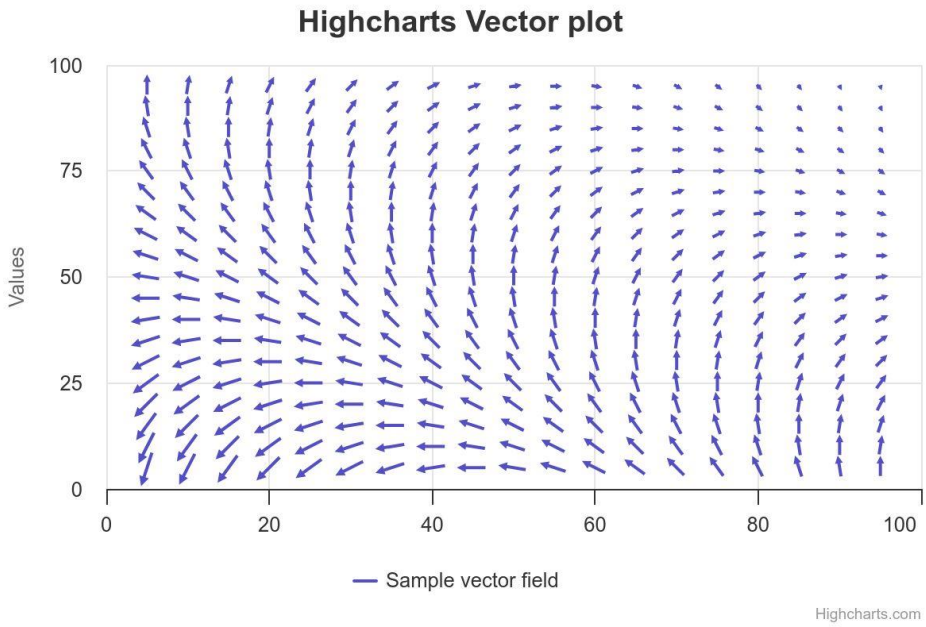
Sl. 3.1 Primjer toplinske karte grada,[19]

- Histogram: Histogram je koristan alat za analizu distribucije intenziteta elektromagnetskog polja. Ovaj grafički prikaz prikazuje učestalost pojavljivanja različitih intenziteta unutar odabranog raspona, što omogućuje istraživačima da brzo i učinkovito identificiraju ključne karakteristike skupa podataka, kao što su modalne vrijednosti i raspon. Osnovni princip histograma je podjela raspona vrijednosti na intervale te prikaz broja podataka koji pripadaju svakom intervalu. U kontekstu analize elektromagnetskog polja, x-os histograma obično predstavlja intenzitet polja, dok y-os prikazuje frekvenciju ili broj mjerenja u svakom intervalu. Histogrami se koriste za identifikaciju dominantnih vrijednosti, a vrhovi u histogramu pokazuju na najčešće intenzitete elektromagnetskog polja. Širina distribucije na histogramu govori koliko su vrijednosti raspršene, pri čemu uska distribucija ukazuje na manje promjene, dok široka distribucija ukazuje na veću varijabilnost analiziranih podataka.



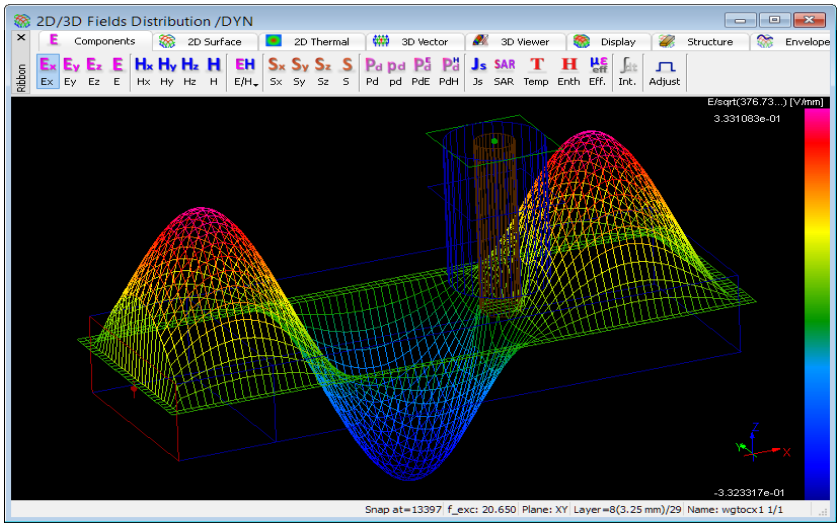
Sl. 3.2 Primjer histograma ocjena studenata,[20]

- Vektorski plotovi: Vektorski plotovi pružaju jasan uvid u dinamiku elektromagnetskog polja. Vizualiziraju vektore u točno određenim točkama prostora. Svaki vektor reprezentira smjer i jačinu polja u svakoj lokaciji, omogućavajući vizualno prepoznavanje kako se polja ponašaju u različitim područjima. Duljina vektora prikazuje jačinu polja, orijentacija vektora prikazuje smjer polja u toj točki. Vektorski plotovi omogućuju vizualizaciju kako se elektromagnetsko polje mijenja kroz prostor. Također kroz niz vektorskih plotova može se vizualizirati promjena elektromagnetskog polja kroz vrijeme.



Sl. 3.3 Primjer testnog vektorskog polja,[21]

- 3D modeliranje: 3D modeliranje koristi se za stvaranje trodimenzionalnih vizualnih prikaza elektromagnetskih polja, omogućavajući korisnicima bolji uvid unutar složenih struktura polja i promatranje iz raznih kutova. Ova tehnika doprinosi boljem razumijevanju prostorne raspodjele i interakcija unutar polja.



Sl. 3.4 Primjer testnog vektorskog polja,[22]

3.2 Vizualizacija podataka u elektromagnetskom polju

Vizualizacija podataka omogućuje transformaciju numeričkih podataka u vizualnu reprezentaciju koja je intuitivnija ljudskom oku. U kontekstu elektromagnetizma, vizualizacija pomaže u identificiranju parametara, poput smjera, intenziteta, međudjelovanja, apsorpcije i refleksije, koji nisu očiti iz numeričkih podatka. Koriste se nekoliko tehnika za vizualizaciju elektromagnetskog polja:

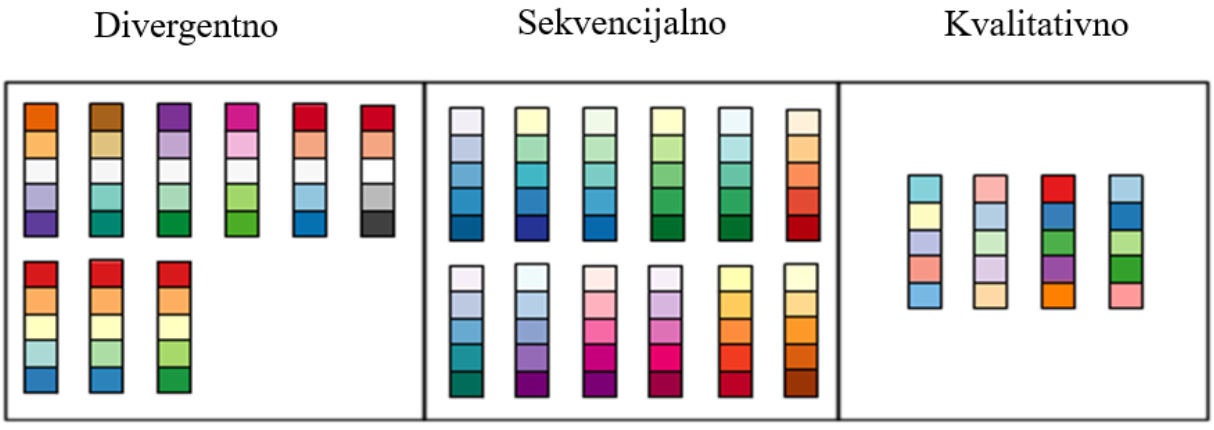
- Silnice ili linije polja: Ova tehnika koristi linije za prikazivanje smjera i magnitude elektromagnetskog polja u prostoru. Linije polja prikazuju jasan uvid u orijentaciju i snagu polja.
- Strujnice: Slično linijama polja, strujnice prikazuju trajektorije čestica pod utjecajem elektromagnetskog polja. Posebno su korisne za vizualizaciju toka energije i smjera propagacije valova
- Konturni dijagrami: Koriste izolinije za prikaz područja konstante magnitude polja. Pomažu u identificiranju regija visokog i niskog intenziteta unutar polja
- Vektorski dijagrami: Vektorski dijagrami koriste strelice za prikaz smjera i intenziteta polja u diskretnim točkama. Daju detaljan uvid u dinamiku elektromagnetskog polja
- Toplinske mape: Toplinske mape koriste boja za prikaz intenziteta polja, gdje toplije boje obično označavaju višu magnitudu. Olakšavaju identificiranje promjena unutar polja na temelju spektra boja

Odabir prikladne vizualizacije ovisi o specifičnom problemu koji se istražuje. Interaktivne i animirane vizualizacije mogu dodatno poboljšati razumijevanje, omogućavajući korisniku manipulaciju prikaza u stvarnom vremenu. Alati poput MATLAB-a, Pythona (s knjižnicama poput Matplotlib) i specijaliziranih softvera poput HFSS i COMSOL često se koriste za generiranje vizualizacija elektromagnetskog polja

3.3 Skale boja

Skale boja pretvaraju kompleksne numeričke vrijednosti u vizualno razumljivu informaciju. Odabir odgovarajuće skale boja omogućava brzu i intuitivnu interpretaciju podataka. Omogućavaju naglašavanje obrazaca i trendova, a u elektromagnetskim poljima omogućavaju izražavanje područja visokog i niskog intenziteta. Ljudsko oko je osjetljivije na varijacije u boji nego na numeričke vrijednosti. Skale boja iskorištavaju ovu sposobnost, omogućavajući preciznu percepciju promjena u elektromagnetskim veličinama. Postoje 3 glavne vrste skala boja koje se koriste u vizualizaciji elektromagnetskih polja, slika 3.5:

- Linearna skala boja: Ova vrsta skale postupno mijenja nijansu (ton) jedne boje kako bi prikazala varijaciju intenziteta, poput temperature ili tlaka. Primjerice može koristiti svijetlo plavu boju za niske vrijednosti postepeno prelazeći u tamniju plavu kako vrijednosti rastu. Prikladna za prikazivanje kvantitativnih podataka, poput jakosti polja ili temperature
- Divergentna skala boja: Koristi se kada podaci imaju centralnu vrijednost s varijacijama u oba smjera. Divergentne skale počinju s dvije različite boje na krajevima spektra, koje se spajaju prema srednjoj neutralnoj boji. Ova se skala često koristi za prikaz devijacija od neke srednje vrijednosti, kao što su temperature ispod ili iznad nekog prosjeka. Također, često se koriste za prikaz pozitivnih i negativnih vrijednosti, poput naboja ili faznog pomaka
- Kvalitativna skala boja: Koristi različite, dobro razlikovane boje kako bi označili kategorije koje nemaju numerički ili prirodni poredak. Kvalitativne skale pomažu u razlikovanju elementa ili grupa bez davanja prednosti bilo kojoj boji. One su korisne za kartiranje kvalitativnih podataka, kao što su različite vrste tla, jezici, ili političke stranke.



Slika 3.5 Vizualizacija različitih vrsta skala boja, [24]

Za generiranje i primjenu skala boja mogu se koristiti sljedeći alati:

- ColorBrewer : Online alat za odabir različitih skala boja
- Matplotlib: Python knjižnica koja nudi ugrađene skale boja
- Paraview: Open-Source aplikacija za vizualizaciju koja podržava napredne tehnike mapiranja boja

4. DIZAJN I IMPLEMENTACIJA ALGORITMA

U okviru ovog završnog rada dizajniran je i implementiran algoritam koji iz digitalne slike električnog polja gdje je jakost polja iskazana preko mape boja, provodi statističku analizu za odabrano područje na slici. Razvijeni algoritam transformira i obrađuje dobivene podatke koristeći sljedeće metode:

- Procesiranje slike i analiza boja: Algoritam mora koristiti tehnike digitalne obrade slika za analizu vizualnih informacija. Korisnik može odabrati specifičnu regiju interesa (ROI) na slici korištenjem grafičkog sučelja.
- Statistička analiza: Za analizu numeričkih podataka primjenjuju se statističke metode. Algoritam koristi NumPy biblioteku za izračunavanje srednje vrijednosti, medijana, standardne devijacije te minimalne i maksimalne vrijednosti. Srednja vrijednost predstavlja prosječnu vrijednost intenziteta elektromagnetskog polja u odabranoj regiji. Medijan predstavlja veličinu od koje je 50% izmjerenih vrijednosti manje, odnosno veće. Standardna devijacija je mjera raspršenosti podataka u odnosu na srednju vrijednost. Minimalna i maksimalna vrijednost daju raspon intenziteta elektromagnetskog polja u analiziranoj regiji.
- Računanje površinske distribucije: Ova metoda odnosi se na proces izračunavanja koliki postotak određene površine slike odgovara određenim kriterijima. Provodi se koristeći tehnike poput segmentacije i klasifikacije unutar slike. Prilikom segmentacije program koristi prethodno definirane vrijednosti dobivene iz analize vrijednosti boja, kako bi rasporedile vrijednosti jakosti električnog polja u različite kategorije. Prilikom klasifikacije svaki element slike unutar odabrane regije klasificira se prema njegovoj pripadajućoj vrijednosti intenziteta. Na temelju klasifikacije program izračunava kumulativnu funkciju distribucije
- Vizualizacija podataka: nakon provedenih proračuna algoritam generira interaktivnu vizualizaciju koje uključuju 3D prikaz, histogram i graf kumulativne distribucije.

4.1. Alati i tehnike za obradu slika

Prilikom obrade slika koriste se alati za analiziranje i manipuliranje boja na slici. Najčešće korišteni alati za obradu slika uključuju :

- MATLAB: Omogućava analizu, dizajn i vizualizaciju fizikalnih veličina koje opisuju elektromagnetska polja. MATLAB sadrži integrirane funkcije za matematičke operacije koje su potrebne za tehničke izračune, što je korisno za analizu elektromagnetskoga polja. Sadrži alate za vizualizaciju i simulaciju matematičkih modela. Prednost korištenja MATLAB-a je omogućen pristup SIMULINK platformi, koja omogućava modeliranje i simulaciju dinamičnih sustava.[25]
- Python: Jedan od najpopularnijih programskih jezika za obradu slika, zbog svoje jednostavnosti i velikog raspona biblioteka. Python se može koristiti u različitim aplikacijama poput web razvoja, analize podataka, umjetna inteligencija i više[26] .Biblioteke poput NumPy omogućavaju rad s matricama, matplotlib omogućava vizualizaciju podataka i SciPy omogućava znanstvene izračune. Glavna prednost Pythona je njegova dostupnost i cijena.

4.2. Koncept i razvoj programskog rješenja

Cilj ovog završnog rada bio je razviti algoritam i napisati program koji omogućava kvantitativnu analizu elektromagnetskog polja kroz analizu mape boja kojima je predstavljeno polje. Metodologija se oslanja na pretvorbu podataka iz digitalne slike u numeričke vrijednosti koje odgovaraju vrijednostima jakosti električnog polja.

Glavna klasa razvijenog programa je **PreciseColorScaleSelector,** koja omogućuje mapiranje boja u numeričke vrijednosti jakosti polja. Korisnici mogu definirati vlastitu skalu boja, što proširuje mogućnosti rada programa. Implementiran je mehanizam za interaktivni odabir regije od interesa na slici. Ovo omogućuje fokusiranu analizu specifičnih područja, što je posebno korisno za detaljno proučavanje pojedinih dijelova polja. Korištenjem Plotly biblioteke algoritam generira set interaktivnih vizualizacija te je uvedeno grafičko korisničko sučelje primjenom PyQt5 biblioteke. Sučelje omogućava jednostavno učitavanje i prikaz slike, odabir regije od interesa, definiranje skale boja i prikaz rezultata analize i vizualizaciju podataka. Algoritam podržava i linearno i logaritamsko skaliranje vrijednosti jakosti polja.

4.3. Programski jezik i razvojno okruženje

Za razvoj algoritma odabran je Python zbog svoje široke podrške u zajednici, velikom broju biblioteka koje su prilagođene potrebnim zahtjevima i njegove cijene. OpenCV (Open Source Computer Vision Library) je odabran za obradu slika zbog svoje jednostavnosti korištenja opcija manipulacija slikama. OpenCV pruža širok raspon funkcija poput funkcija za detekciju rubova, transformaciju boja, i odabir ROI (*Region of Interest* – regija interesa). Za izradu grafičkog korisničkog sučelja, odabran je PyQt5. PyQt5 omogućava brz i jednostavan razvoj računalnih aplikacija koje zahtijevaju korisničku interakciju, olakšavajući proces odabira slika. Plotly se koristi za vizualizaciju podataka, omogućavajući stvaranje interaktivnih, dinamičkih toplinskih karata koje detaljno prikazuju varijacije jakosti ispitivanog polja. Numpy i SciPy su biblioteke koje se služe za obradu brojeva, obično korištene za manipulaciju i analizu podataka. Omogućuju matematičke transformacija potrebne u zadatku. Pomoću opisanih biblioteka, moguće je stvoriti prototip aplikacije koja je jednostavna za korištenje i prilagodljiva za buduća korištenja.

4.4. Detalji implementacije

Razvoj algoritma za analizu elektromagnetskog polja temeljen na analizi skale boja obuhvaća implementaciju funkcija kako slijedi.

Učitavanje i obrada slike - Algoritam započinje učitavanjem slike sa željene lokacije na računalu. Ovo se postiže korištenjem cv2.imread() funkcije iz OpenCV biblioteke. Primjer implementacije prikazan je na slici 4.1

```
def load_image(file_path):
    image = cv2.imread(file_path)
    if image is None:
        raise ValueError("Nije moguće učitati sliku.")
    return image
```

Sl. 4.1 Primjer implementacije cv2.imread() funkcije.

Nakon učitavanja, slika se konvertira iz BGR (standardni format u OpenCV-u) u RGB format radi lakše manipulacije i analize. Ovo se postiže korištenjem cv2.cvtColor() funkcije vidljivo na slici 4.2

```
def convert_to_rgb(image):
    return cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

Sl. 4.2 Implementacija cv2.cvtColor() funkcije.

Odabir regije interesa - Odabir ROI-a implementiran je korištenjem QRubberBand klase iz PyQt5, koja omogućuje korisniku da odabere pravokutnik preko željenog dijela slike (slika 4.3)

```
class ImageAnalysisGUI(QMainWindow):
    def __init__(self):
        super().__init__()
        self.rubberBand = None
        self.origin = None

    def mousePressEvent(self, event):
        self.origin = event.pos()
        self.rubberBand = QRubberBand(QRubberBand.Rectangle, self)
        self.rubberBand.setGeometry(QRect(self.origin, QSize()))
        self.rubberBand.show()

    def mouseMoveEvent(self, event):
        if self.rubberBand:
            self.rubberBand.setGeometry(QRect(self.origin, event.pos()).normalized())

    def mouseReleaseEvent(self, event):
        if self.rubberBand:
            self.selected_region = self.get_selection_coordinates()
            self.rubberBand.hide()
```

Sl. 4.3 Implementacije ROI-a.

Mapiranje boja na vrijednosti jakosti električnog/magnetskog polja - Korisnik prvo odabire vertikalnu liniju na slici koja predstavlja skalu boja, zatim definira minimalnu i maksimalnu vrijednost jakosti elektromagnetskog polja koje odgovaraju krajnjim točkama skale boja (slika 4.4).

```python
def assign_values_to_colors(unique_colors, min_value, max_value, scale_type):
    num_colors = len(unique_colors)
    if scale_type == 'Linearna':
        values = np.linspace(min_value, max_value, num_colors)
    else:  # Logaritamska
        values = np.geomspace(max(min_value, 1e-6), max_value, num_colors)
    return dict(zip(map(tuple, unique_colors), values))
```

Sl. 4.4 implementacije minimalne i maksimalne vrijednosti.

Analiza regije interesa - Nakon što su boje mapirane na vrijednosti iskazane u skali boja, algoritam analizira odabranu regiju interesa. Svaki element slike u regiji interesa mapira se na odgovarajuću vrijednost jakosti polja, nakon čega se provodi statistička analiza, vidljivo na slikama 4.5,4.6

```python
def analyze_region(region, color_tree, color_to_value):
    height, width, _ = region.shape
    values = np.zeros((height, width))
    for i in range(height):
        for j in range(width):
            color = tuple(region[i, j])
            dist, idx = color_tree.query(color)
            closest_color = tuple(color_tree.data[idx])
            values[i, j] = color_to_value.get(closest_color, np.nan)
    return values
```

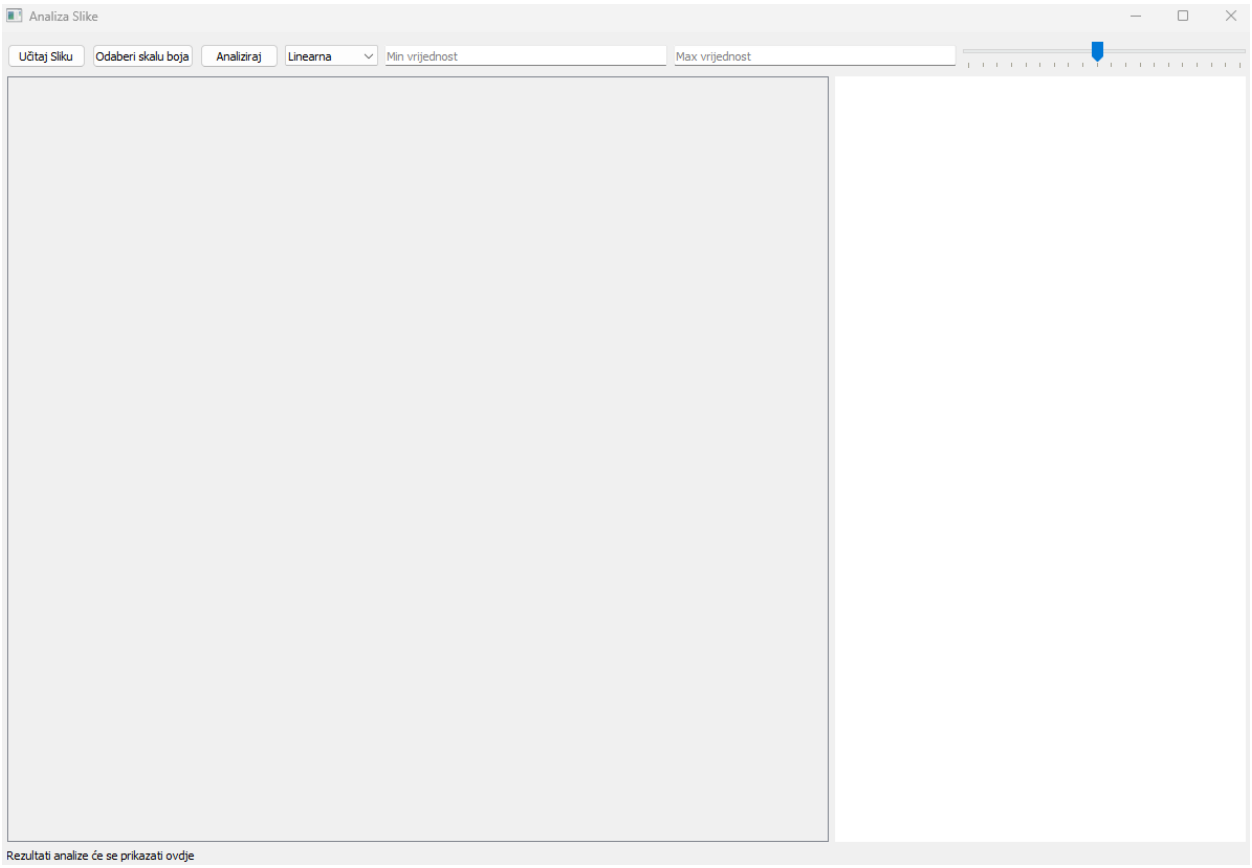
Sl. 4.5 Mapiranje vrijednosti piksela.

```python
def calculate_statistics(values):
    valid_values = values[~np.isnan(values)]
    return {
        'mean': np.mean(valid_values),
        'median': np.median(valid_values),
        'std': np.std(valid_values),
        'min': np.min(valid_values),
        'max': np.max(valid_values)
    }
```

Sl. 4.6 Statistička analiza.

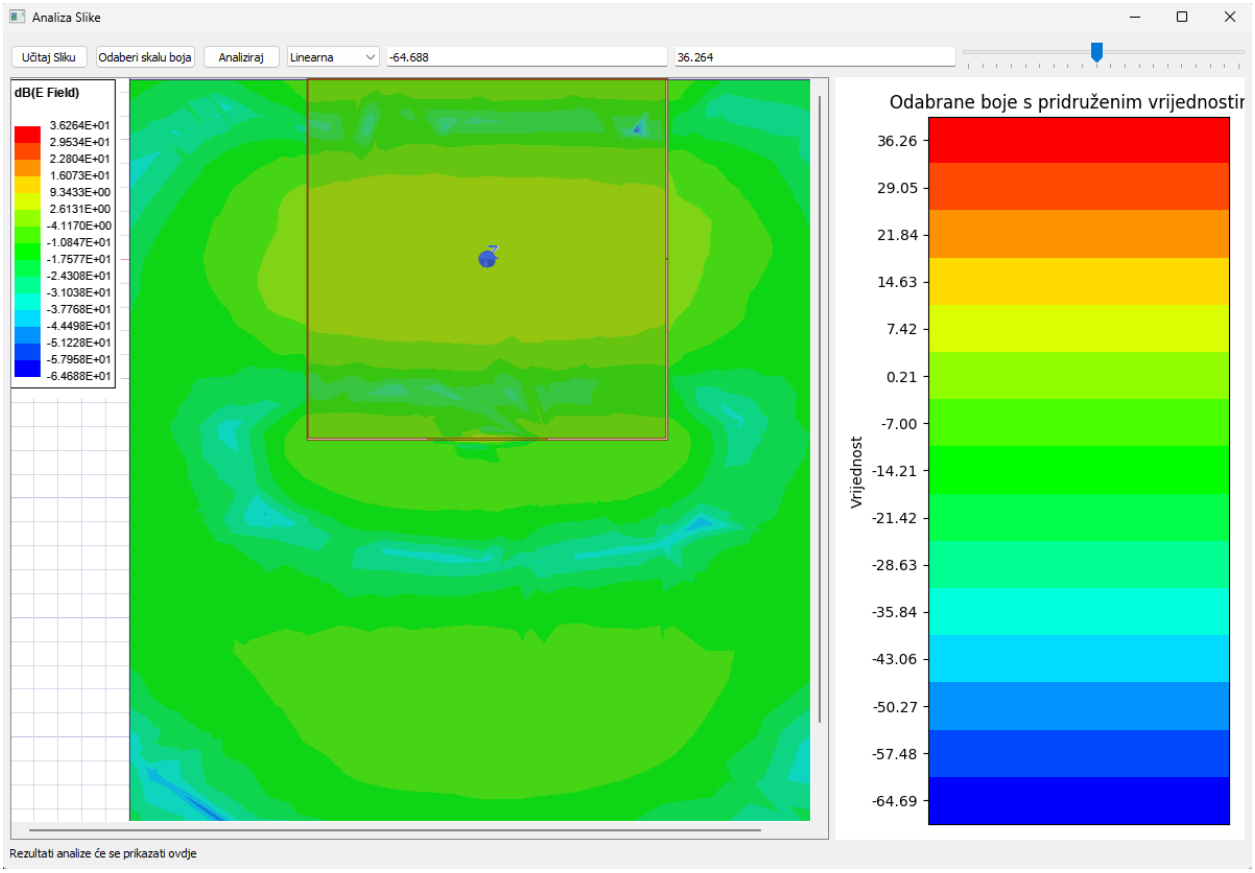
5. PREGLED KORIŠTENJA PROGRAMA

Program se pokreće putem naredbenog retka. Glavno sučelje programa će se otvoriti, prikazujući prostor za učitavanje slike, prostor gdje će se prikazati vrijednosti analize, dugmad za upravljanje programom i prostor za unos minimalne i maksimalne vrijednosti , prikazano na slici 5.1



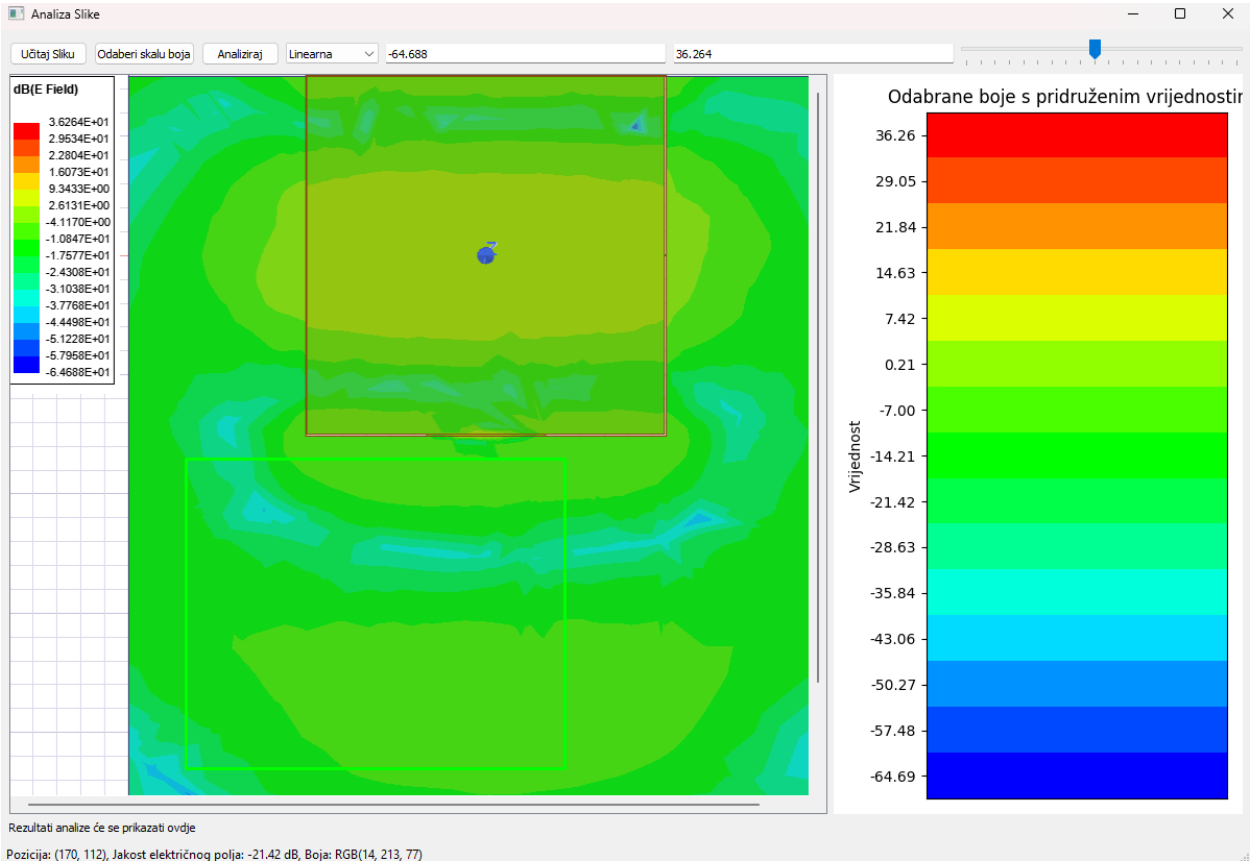
Sl. 5.1 Glavno sučelje programa

Korisnik prvo unosi željenu sliku pritiskom na dugme "Učitaj sliku". Nakon toga unosi vrijednosti skale boja te pritišće dugme "Odaberi skalu boja". Program daje mogućnost korisniku odabir skale boja povlačenjem linije na slici na području gdje se skala boja nalazi(slika 5.2). Nadalje, korisnik treba definirati radi li se o linearnoj i logaritamskoj skali boja.



Sl. 5.2 Prikaz glavnog sučelja nakon uspješnog odabira regije.

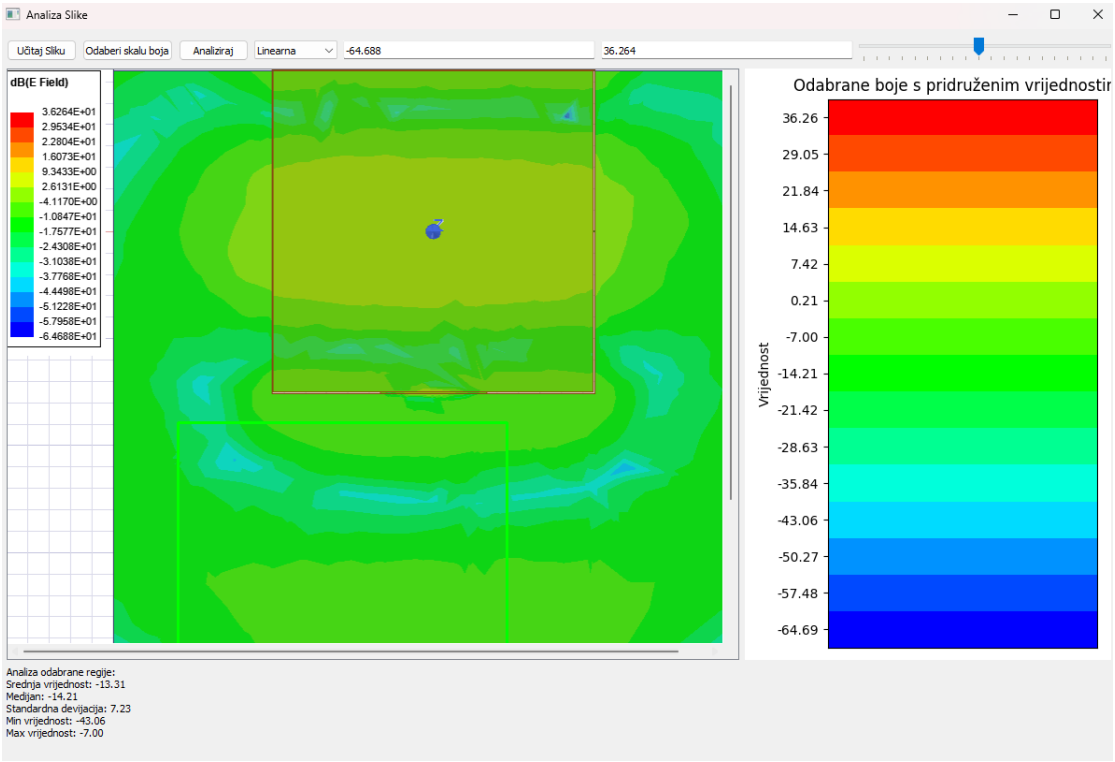
Korisnik može odabrati područje od interesa klikom i povlačenjem miša preko željenog područja. Odabrana regija će biti označena zelenim pravokutnikom vidljivo na slici 5.3. Prelaženjem mišem preko određenog elementa slike unutar odabrane regije, program će na dnu prozora prikazati vrijednost jakosti električnog polja na tom mjestu. Kako bi se korisniku olakšao odabir određene regije, u gornjem desnom kutu programa nalazi se povećalo.



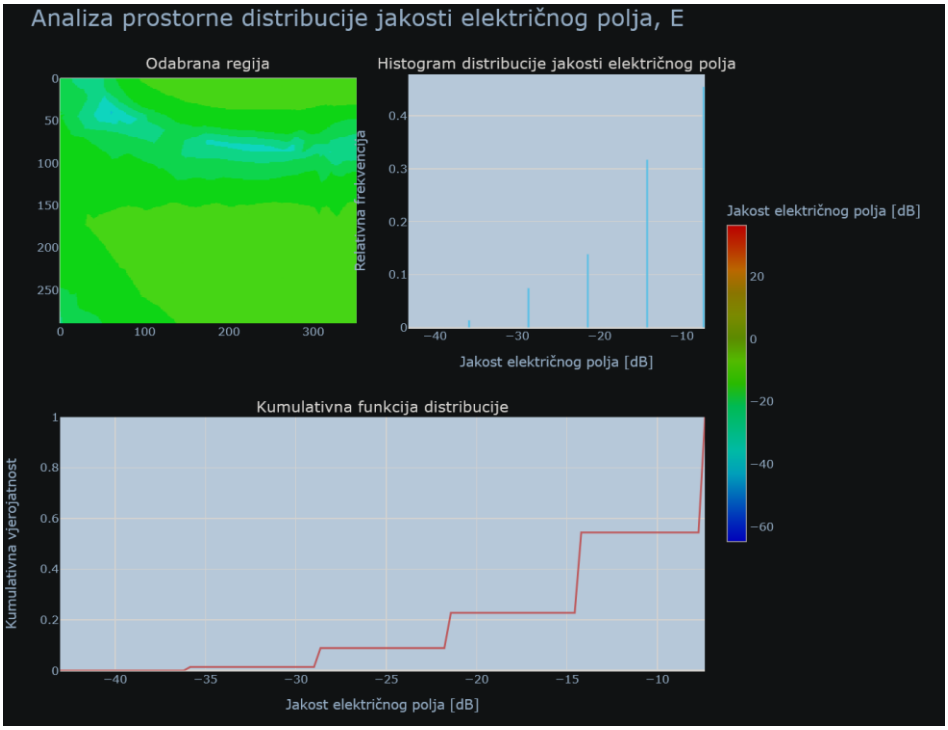
Sl. 5.3 Prikaz glavnog sučelja nakon odabira željene regije.

Pritiskom na tipku "Analiziraj", program će izvršiti analizu odabrane regije koristeći definiranu skalu boja. Rezultati analize će se prikazati u donjem dijelu sučelja vidljivo na slici 5.4, uključujući srednju vrijednost, medijan, standardnu devijaciju, minimalnu i maksimalnu vrijednost. Nakon analize, program će otvoriti novi prozor s četiri grafička prikaza (slika 5.5):

- Odabrana regija
- 3D prikaz vrijednosti u odabranoj regiji
- Histogram distribucije vrijednosti
- Kumulativna distribucija vrijednosti



Sl. 5.4 Prikaz glavnog sučelja nakon analiziranja rezultata.



Sl. 5.5 Grafički prikaz odabrane regije i rezultata analize.

6. ZAKLJUČAK

Elektromagnetsko polje je fundamentalan koncept u fizici koji opisuje interakciju električnih naboja. Ono se sastoji od električnog polja, koje nastaje oko električnih naboja, i magnetskog polja, koje se stvara oko magneta i električnih struja. Za proračun vrijednosti jakosti električnog ili magnetskog polja složenijih struktura koriste se programi koji imaju implementirane numeričke metode poput metode momenata ili metode konačnih elemenata. Rezultati proračuna jakosti polja obično se vizualiziraju primjenom mapa boja, s pridruženom skalom koja mapira vrijednosti jakosti polja na izabrane boje. U ovom je radu fokus bio na stvaranju programskog rješenja koje će iz već stvorenih digitalnih slika raspodjele vrijednosti vektora jakosti električnog polja provesti određenu statističku analizu. Programski kod razvijen u Python programskom jeziku, uz korištenje specijaliziranih biblioteka poput NumPy, OpenCV, PyQt5 i Plotly, daje mogućnost naknadne statističke analize polja za odabrana područja od interesa na slici polja.

6. LITERATURA

[1] Halliday, D., Resnick, R., Walker, J., Fundamentals of Physics, 10th Edition, Wiley, 2013, str. 610.

[2] GeeksforGeeks, "Basic Properties of Electric Charge," dostupno na: https://www.geeksforgeeks.org/basic-properties-of-electric-charge/, [datum pristupa: 16. lipnja 2024.]

[3] M. Chen, "Coulomb's Law", Phys102 Lecture Notes, Simon Fraser University, dostupno na: http://www.sfu.ca/~mxchen/phys1021003/P102LN02.pdf, [datum pristupa: 16. lipnja 2024.]

[4] D. Halliday, R. Resnick, J. Walker, *Fundamentals of Physics*, 10th Edition, Wiley, 2013, str. 630.

[5] "Magnetic field lines formed inside a solenoid," ResearchGate, dostupno na: https://www.researchgate.net/figure/Magnetic-field-lines-formed-inside-a-solenoid_fig1_343274593, [datum pristupa: 16. lipnja 2024.

[6] LibreTexts, "Electromagnetic Radiation [online]," LibreTexts Chemistry Library, dostupno na: https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/Supplemental_Modules_(Physical_and_Theoretical_Chemistry)/Spectroscopy/Fundamentals_of_Spectroscopy/Electromagnetic_Radiation, 2020, [Pristup 12.4.2024].

[7] Generalić, E., "Spektar elektromagnetskog zračenja," dostupno na: https://glossary.periodni.com/glosar.php?hr=spektar+elektromagnetskog+zra%C4%8Denja, [datum pristupa: 10. rujna 2024.].

[8] Griffiths, D. J. (2013). *Introduction to Electrodynamics* (4th ed.). Pearson, str. 337.

[9]Owlcation, "Maxwell's Equations and Displacement Current," dostupno na: https://owlcation.com/stem/maxwell-equations-displacement-current, [datum pristupa: 10. rujna 2024.].

[10] Internshala, "What is ANSYS?," dostupno na: https://trainings.internshala.com/blog/what-is-ansys/, [datum pristupa: 16. lipnja 2024.]

[11] Y. Liu, "A Review of the Influence of Electromagnetic Fields on Cellulose Synthesis," *Open Journal of Organic Polymer Materials*, vol. 7, no. 4, pp. 71-78, 2017, dostupno na: https://www.scirp.org/journal/paperinformation?paperid=79007, [datum pristupa: 16. lipnja 2024.]

[12] COMSOL, "COMSOL Multiphysics® Software," dostupno na: https://www.comsol.com/, [datum pristupa: 16. lipnja 2024.]

[13] E. A. Fattah, F. K. Abdulqader, "Electromagnetic and Thermal Simulations of Human Neurons for SAR Applications," *Open Journal of Antennas and Propagation*, vol. 4, no. 1, str. 27-36, 2016, dostupno na: https://www.researchgate.net/publication/306085421, [datum pristupa: 16. lipnja 2024.]

[14] 3DS, "SIMULIA CST Studio Suite," dostupno na: https://www.3ds.com/products/simulia/cst-studio-suite, [datum pristupa: 16. lipnja 2024.]

[15] A. Saito, T. Matsui, "Microwave Microscopy Applied to EMC Problem: Visualisation of Electromagnetic Field in the Vicinity of Electronic Circuit and Effect of Nanomaterial Coating," *Open Journal of Antennas and Propagation*, vol. 5, no. 2, str. 45-54, 2017, dostupno na: https://www.researchgate.net/publication/317120494, [datum pristupa: 16. lipnja 2024.]

[16] N. J. Fliege, "Super-NEC: Program za simulaciju antena i unutarnjeg širenja," dostupno na: ResearchGate, [datum pristupa: 16. lipnja 2024.].

[17] C. D. Nikolopoulos, A. T. Baklezos, C. N. Capsalis, "Auto-reconfigurable Patch Antenna for Biomedical Single-Channel Multi-Frequency Microwave Radiometry Applications," *Progress In Electromagnetics Research C*, dostupno na: ResearchGate, [datum pristupa: 16. lipnja 2024.]

[18]Autor(i), "Contour plot of the electric field equipotential lines and vector field plot of the electric field," dostupno na: https://www.researchgate.net/figure/a-Contour-plot-of-the-electric-field-equipotential-lines-b-Vector-field-plot-of-the_fig6_275621766, [datum pristupa: 16. lipnja 2024.].

[19] Atlassian, Heatmap: The Complete Guide [online], Atlassian, United States, 2024, dostupno na: https://www.atlassian.com/data/charts/heatmap-complete-guide [Pristup 18.4.2024].

[20] Spotfire, What is a Histogram Chart? [online], Spotfire, United States, 2024, dostupno na: https://www.spotfire.com/glossary/what-is-a-histogram-chart [Pristup 18.4.2024].

[21] Highcharts, What is a Vector Plot? [online], Highcharts, 2024, dostupno na: https://www.highcharts.com/docs/chart-and-series-types/vector-plot [Pristup 9.6.2024].

[22]QW-ED, EM Fields in Microwave Electronics [online], QW-ED, dostupno na: https://www.qwed.com.pl/app/app_emfields.html [Pristup 9.6.2024].

[24] Faculty of Science, Charles University, "Colors [online]," Charles University, available at: https://web.natur.cuni.cz/~langhamr/lectures/vtfg1/mapinfo_2/barvy/colors.html, 2024, [Pristup 12.4.2024].

[25] "MATLAB i Simulink [online]," MathWorks, https://www.mathworks.com, 2024, dostupno na: https://www.mathworks.com [Pristup 18.4.2024].

[26] Python Software Foundation, "Python [online]," Python Software Foundation, dostupno na: https://www.python.org, 2024, [Pristup 18.4.2024].

SAŽETAK

Ovaj završni rad opisuje razvoj i implementacija algoritma za analizu elektromagnetskog polja pomoću obrade digitalnih slika nastalih vizualizacijom raspodjele jakosti električnog polja proračunatom nekim od specijaliziranih programa. Rad obuhvaća opis ključnih komponenti algoritma, uključujući precizno mapiranje boja, interaktivnu selekciju regije od interesa, statističku analizu i vizualizaciju rezultata. Opisane su prednosti korištenja Python programskog jezika i specijaliziranih biblioteka poput NumPy, OpenCV, PyQt5 i Plotly u razvoju ovog rješenja

Ključne riječi: Elektromagnetsko polje, obrada digitalnih slika, mapiranje boja, kvantitativna analiza, Python, vizualizacija podataka, grafičko korisničko sučelje

ABSTRACT

This final thesis describes the development and implementation of an algorithm for electromagnetic field analysis using digital image processing of images created by visualizing the electric field strength distribution calculated by specialized programs. The work includes a description of key components of the algorithm, including precise color mapping, interactive selection of the region of interest, statistical analysis, and result visualization. The advantages of using the Python programming language and specialized libraries such as NumPy, OpenCV, PyQt5, and Plotly in developing this solution are described.

Keywords: Electromagnetic field, digital image processing, color mapping, quantitative analysis, Python, data visualization, graphical user interface

PRILOZI

```
import sys
import cv2
import numpy as np
from PyQt5.QtWidgets import (QApplication, QMainWindow, QLabel, QPushButton, QVBoxLayout, QHBoxLayout,
                             QWidget, QFileDialog, QComboBox, QLineEdit, QRubberBand, QMessageBox,
                             QScrollArea, QSlider)
from PyQt5.QtGui import QPixmap, QImage
from PyQt5.QtCore import Qt, QRect, QPoint
import matplotlib.pyplot as plt
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from scipy.spatial import KDTree
import logging

logging.basicConfig(level=logging.DEBUG)

class PreciseColorScaleSelector:
    def __init__(self):
        self.color_map = None
        self.min_value = None
        self.max_value = None
        self.unique_colors = None
        self.values = None
        self.color_to_value = None
        self.color_tree = None

    def select_color_scale(self, image, x, y1, y2):
        logging.debug(f"Selecting color scale from image shape {image.shape}, x={x}, y1={y1}, y2={y2}")
        try:
            color_scale = image[y1:y2, x:x+1]
            logging.debug(f"Color scale shape: {color_scale.shape}")

            if len(color_scale.shape) == 2:
                color_scale_rgb = cv2.cvtColor(color_scale, cv2.COLOR_GRAY2RGB)
            elif color_scale.shape[2] == 3:
                color_scale_rgb = cv2.cvtColor(color_scale, cv2.COLOR_BGR2RGB)
            else:
                raise ValueError(f"Unexpected color scale shape: {color_scale.shape}")

            unique_colors = np.unique(color_scale_rgb.reshape(-1, 3), axis=0)
            logging.debug(f"Number of unique colors: {len(unique_colors)}")

            if len(unique_colors) == 0:
                raise ValueError("No unique colors found in the selected scale")

            y_positions = [np.where((color_scale_rgb == color).all(axis=2))[0][0] for color in unique_colors]
            sorted_indices = np.argsort(y_positions)[::-1]

            self.unique_colors = unique_colors[sorted_indices]
            logging.debug(f"Sorted unique colors: {self.unique_colors}")
            return [tuple(color) for color in self.unique_colors]
        except Exception as e:
            logging.exception("Error in select_color_scale")
            raise

    def assign_values_to_colors(self, min_value, max_value, scale_type):
        logging.debug(f"Assigning values to colors: min={min_value}, max={max_value}, scale_type={scale_type}")
        try:
            self.min_value, self.max_value = min_value, max_value
            num_colors = len(self.unique_colors)

            if num_colors < 2:
```

```
            raise ValueError(f"Not enough unique colors: {num_colors}")

        if scale_type == 'Linearna':
            self.values = np.linspace(min_value, max_value, num_colors)
        else:
            self.values = np.geomspace(max(min_value, 1e-6), max_value, num_colors)

        self.color_to_value = dict(zip(map(tuple, self.unique_colors), self.values))
        self.color_tree = KDTree(self.unique_colors)
        logging.debug(f"Color map created with {len(self.color_to_value)} entries")

        for color, value in zip(self.unique_colors, self.values):
            logging.debug(f"Color RGB{tuple(color)} mapped to value {value}")
    except Exception as e:
        logging.exception("Error in assign_values_to_colors")
        raise

def get_color_value(self, color):
    try:
        dist, idx = self.color_tree.query(color)
        closest_color = tuple(self.unique_colors[idx])
        value = self.color_to_value.get(closest_color, np.nan)
        if np.isnan(value):
            logging.debug(f"No value found for color: {color}")
        return value
    except Exception as e:
        logging.exception(f"Error in get_color_value for color {color}")
        return np.nan

def analyze_region(self, region):
    if self.color_to_value is None:
        logging.warning("Color to value map is None in analyze_region")
        return None

    try:
        region_rgb = cv2.cvtColor(region, cv2.COLOR_BGR2RGB)
        height, width, _ = region_rgb.shape
        logging.debug(f"Analyzing region with shape: {region_rgb.shape}")

        values = np.zeros((height, width))
        for i in range(height):
            for j in range(width):
                color = tuple(region_rgb[i, j])
                value = self.get_color_value(color)
                if np.isnan(value):
                    logging.debug(f"NaN value for color: {color}")
                values[i, j] = value

        logging.debug(f"Mapped values shape: {values.shape}")
        logging.debug(f"Mapped values range: {np.nanmin(values)} - {np.nanmax(values)}")

        valid_values = values[~np.isnan(values)]
        logging.debug(f"Number of valid values: {len(valid_values)}")

        if len(valid_values) == 0:
            logging.warning("No valid values found in analyze_region")
            return None

        result = {
            'mean': np.mean(valid_values),
            'median': np.median(valid_values),
            'std': np.std(valid_values),
            'min': np.min(valid_values),
            'max': np.max(valid_values),
            'values': values,
            'original_region': region
        }
```

```
            logging.debug(f"Analysis results: {result}")
            return result
        except Exception as e:
            logging.exception(f"Error in analyze_region: {str(e)}")
            return None

class ImageAnalysisGUI(QMainWindow):
    def __init__(self):
        super().__init__()
        self.initUI()
        self.rubberBand = None
        self.origin = None
        self.current = None
        self.image = None
        self.display_img = None
        self.color_selector = PreciseColorScaleSelector()
        self.selection_mode = 'region'
        self.selected_region = None
        self.zoom_factor = 1.0
        self.field_strength_unit = "dB"
    def initUI(self):
        self.setWindowTitle('Analiza Slike')
        self.setGeometry(100, 100, 1200, 800)

        central_widget = QWidget()
        main_layout = QVBoxLayout()

        top_layout = QHBoxLayout()

        buttons = [
            ('Učitaj Sliku', self.load_image),
            ('Odaberi skalu boja', self.start_color_scale_selection),
            ('Analiziraj', self.analyze_image)
        ]

        for text, func in buttons:
            button = QPushButton(text)
            button.clicked.connect(func)
            top_layout.addWidget(button)

        self.scale_type = QComboBox()
        self.scale_type.addItems(['Linearna', 'Logaritamska'])
        top_layout.addWidget(self.scale_type)

        for placeholder in ['Min vrijednost', 'Max vrijednost']:
            line_edit = QLineEdit()
            line_edit.setPlaceholderText(placeholder)
            setattr(self, placeholder.split()[0].lower() + '_value', line_edit)
            top_layout.addWidget(line_edit)

        self.zoom_slider = QSlider(Qt.Horizontal)
        self.zoom_slider.setRange(10, 200)
        self.zoom_slider.setValue(100)
        self.zoom_slider.setTickInterval(10)
        self.zoom_slider.setTickPosition(QSlider.TicksBelow)
        self.zoom_slider.valueChanged.connect(self.zoom_image)
        top_layout.addWidget(self.zoom_slider)

        main_layout.addLayout(top_layout)

        middle_layout = QHBoxLayout()

        self.scroll_area = QScrollArea()
        self.image_label = QLabel()
        self.image_label.setAlignment(Qt.AlignCenter)
        self.image_label.setMouseTracking(True)
        self.image_label.mousePressEvent = self.on_image_mouse_press
```

```
        self.image_label.mouseMoveEvent = self.on_image_mouse_move
        self.image_label.mouseReleaseEvent = self.on_image_mouse_release
        self.scroll_area.setWidget(self.image_label)
        self.scroll_area.setWidgetResizable(True)
        middle_layout.addWidget(self.scroll_area, 2)

        self.figure = plt.figure(figsize=(5, 8))
        self.canvas = FigureCanvas(self.figure)
        middle_layout.addWidget(self.canvas, 1)

        main_layout.addLayout(middle_layout)

        self.results_label = QLabel('Rezultati analize će se prikazati ovdje')
        main_layout.addWidget(self.results_label)

        central_widget.setLayout(main_layout)
        self.setCentralWidget(central_widget)

    def load_image(self):
        file_name, _ = QFileDialog.getOpenFileName(self, "Odaberi Sliku", "", "Image Files (*.png *.jpg *.bmp)")
        if file_name:
            self.image = cv2.imread(file_name)
            if self.image is None:
                QMessageBox.critical(self, "Greška", "Nije moguće učitati sliku.")
                return
            self.display_img = self.image.copy()
            self.display_image()
            self.selected_region = None

    def display_image(self):
        if self.display_img is None or self.display_img.size == 0:
            return

        height, width, channel = self.display_img.shape
        bytes_per_line = 3 * width
        q_image = QImage(self.display_img.data, width, height, bytes_per_line, QImage.Format_RGB888).rgbSwapped()
        pixmap = QPixmap.fromImage(q_image)
        scaled_pixmap = pixmap.scaled(pixmap.size() * self.zoom_factor, Qt.KeepAspectRatio, Qt.SmoothTransformation)
        self.image_label.setPixmap(scaled_pixmap)
        self.image_label.resize(scaled_pixmap.size())

    def start_color_scale_selection(self):
        if self.image is None:
            QMessageBox.warning(self, "Upozorenje", "Prvo učitajte sliku.")
            return
        self.selection_mode = 'color_scale'
        QMessageBox.information(self, "Odabir skale boja", "Odaberite vertikalnu liniju koja predstavlja skalu boja na slici.")

    def on_image_mouse_press(self, event):
        if self.image is None:
            return
        self.origin = event.pos()
        self.rubberBand = QRubberBand(QRubberBand.Rectangle if self.selection_mode == 'region' else QRubberBand.Line,
self.image_label)
        self.rubberBand.setGeometry(QRect(self.origin, QPoint(self.origin.x(), self.origin.y())))
        self.rubberBand.show()

    def on_image_mouse_move(self, event):
        if self.rubberBand:
            self.current = QPoint(self.origin.x(), event.pos().y()) if self.selection_mode == 'color_scale' else event.pos()
            self.rubberBand.setGeometry(QRect(self.origin, self.current).normalized())
        self.show_pixel_value(event.pos())

    def on_image_mouse_release(self, event):
        if self.rubberBand:
            self.rubberBand.hide()
            if self.selection_mode == 'color_scale':
```

```
            self.process_color_selection()
        else:
            self.process_region_selection()

    def process_color_selection(self):
        logging.debug("Starting process_color_selection")
        if self.image is None or self.origin is None or self.current is None:
            logging.warning("Image or selection points are None")
            return

        try:
            min_value = float(self.min_value.text().strip())
            max_value = float(self.max_value.text().strip())

            if min_value >= max_value:
                raise ValueError("Minimalna vrijednost mora biti manja od maksimalne.")

        except ValueError as e:
            logging.error(f"Invalid input values: {str(e)}")
            QMessageBox.warning(self, "Upozorenje", f"Nevažeće vrijednosti: {str(e)}\nMolimo unesite valjane brojeve.")
            return

        scale_x = self.image.shape[1] / self.image_label.width()
        scale_y = self.image.shape[0] / self.image_label.height()

        x = int(self.origin.x() * scale_x)
        y1 = int(min(self.origin.y(), self.current.y()) * scale_y)
        y2 = int(max(self.origin.y(), self.current.y()) * scale_y)

        logging.debug(f"Selected color scale region: x={x}, y1={y1}, y2={y2}")

        try:
            unique_colors = self.color_selector.select_color_scale(self.image, x, y1, y2)

            if not unique_colors:
                raise ValueError("Nije moguće odabrati boje. Molimo pokušajte ponovno.")

            logging.debug(f"Number of unique colors selected: {len(unique_colors)}")

            self.color_selector.assign_values_to_colors(min_value, max_value, self.scale_type.currentText())

            if self.color_selector.color_to_value is None:
                raise ValueError("Skala boja nije definirana")

            logging.debug("Selected colors and values:")
            for color, value in zip(self.color_selector.unique_colors, self.color_selector.values):
                logging.debug(f"Color: {color}, Value: {value}")

            self.visualize_discrete_colors(unique_colors)
            QMessageBox.information(self, "Odabir boja", f"Uspješno odabrano {len(unique_colors)} jedinstvenih boja.")
            self.selection_mode = 'region'
        except Exception as e:
            logging.exception("Error in process_color_selection")
            QMessageBox.critical(self, "Greška", f"Došlo je do greške pri odabiru skale boja: {str(e)}")

    def process_region_selection(self):
        if self.image is None or self.origin is None or self.current is None:
            return

        x1, y1, x2, y2 = self.get_selection_coordinates()
        self.selected_region = (x1, y1, x2, y2)
        self.display_img = self.image.copy()
        cv2.rectangle(self.display_img, (x1, y1), (x2, y2), (0, 255, 0), 2)
        self.display_image()

    def get_selection_coordinates(self):
        scale_x = self.image.shape[1] / self.image_label.width()
```

```
    scale_y = self.image.shape[0] / self.image_label.height()
    x1 = max(0, int(min(self.origin.x(), self.current.x()) * scale_x))
    y1 = max(0, int(min(self.origin.y(), self.current.y()) * scale_y))
    x2 = min(self.image.shape[1], int(max(self.origin.x(), self.current.x()) * scale_x))
    y2 = min(self.image.shape[0], int(max(self.origin.y(), self.current.y()) * scale_y))
    return x1, y1, x2, y2

def visualize_discrete_colors(self, unique_colors):
    if not unique_colors:
        return

    try:
        self.figure.clear()
        ax = self.figure.add_subplot(111)


        rgb_colors = unique_colors
        num_colors = len(rgb_colors)
        color_display = np.array(rgb_colors).astype(np.uint8)
        color_display = np.repeat(color_display[:, np.newaxis, :], 50, axis=1)

        ax.imshow(color_display, aspect='auto', extent=[0, 1, 0, num_colors], origin='lower')

        y_ticks = np.arange(0.5, num_colors, 1)
        ax.set_yticks(y_ticks)

        if hasattr(self.color_selector, 'values') and self.color_selector.values is not None:

            values = self.color_selector.values
            ax.set_yticklabels([f"{v:.2f}" for v in values])
        else:
            ax.set_yticklabels([])
            logging.warning("Vrijednosti nisu dostupne za prikaz na y-osi.")

        ax.set_xticks([])

        ax.set_title("Odabrane boje s pridruženim vrijednostima")
        ax.set_ylabel("Vrijednost")

        ax.yaxis.set_label_position("left")
        ax.yaxis.tick_left()

        plt.tight_layout()
        self.canvas.draw()
    except Exception as e:
        logging.error(f"Error in visualize_discrete_colors: {str(e)}")
        QMessageBox.warning(self, "Upozorenje", f"Greška pri vizualizaciji boja: {str(e)}")

def analyze_image(self):
    try:
        logging.debug("Starting image analysis")

        if self.image is None:
            raise ValueError("Slika nije učitana")

        if self.selected_region is None:
            raise ValueError("Regija za analizu nije odabrana")

        if self.color_selector.color_to_value is None:
            raise ValueError("Skala boja nije definirana")

        x1, y1, x2, y2 = self.selected_region
        logging.debug(f"Selected region: {self.selected_region}")

        if x1 >= x2 or y1 >= y2:
            raise ValueError("Nevažeća regija za analizu")
```

```
        region = self.image[y1:y2, x1:x2]
        logging.debug(f"Region shape: {region.shape}")

        if region.size == 0:
            raise ValueError("Odabrana regija je prazna")

        results = self.color_selector.analyze_region(region)
        logging.debug(f"Analysis results: {results}")

        if results is None:
            raise ValueError("Analiza nije uspjela - nema rezultata")

        if 'values' not in results:
            raise ValueError("Analiza nije uspjela - nedostaju mapirane vrijednosti")

        if np.all(np.isnan(results['values'])):
            raise ValueError("Sve mapirane vrijednosti su NaN")

        results_text = (f'Analiza odabrane regije:\n'
                        f'Srednja vrijednost: {results["mean"]:.2f}\n'
                        f'Medijan: {results["median"]:.2f}\n'
                        f'Standardna devijacija: {results["std"]:.2f}\n'
                        f'Min vrijednost: {results["min"]:.2f}\n'
                        f'Max vrijednost: {results["max"]:.2f}\n')

        self.results_label.setText(results_text)
        self.visualize_results(region, results)
        logging.debug("Analysis completed successfully")

    except Exception as e:
        logging.exception("Error during image analysis")
        QMessageBox.critical(self, "Greška", f"Došlo je do greške tijekom analize: {str(e)}")

def visualize_results(self, region, results):
    fig = make_subplots(rows=2, cols=2,
                        subplot_titles=("Odabrana regija", "Histogram distribucije jakosti električnog polja",
                                        "Kumulativna funkcija distribucije"),
                        specs=[[{"type": "image"}, {"type": "histogram"}],
                           [{"type": "scatter", "colspan": 2}, None]],
                        vertical_spacing=0.15,
                        horizontal_spacing=0.08)

    fig.add_trace(go.Image(z=cv2.cvtColor(region, cv2.COLOR_BGR2RGB)), row=1, col=1)

    hist_data = results['values'].flatten()
    hist_data = hist_data[~np.isnan(hist_data)]


    hist, bin_edges = np.histogram(hist_data, bins=100)
    total_pixels = len(hist_data)
    relative_freq = hist / total_pixels

    fig.add_trace(go.Bar(x=bin_edges[:-1], y=relative_freq, name="Histogram",
                         marker_color='rgba(0, 150, 200, 0.7)'), row=1, col=2)


    cdf = np.cumsum(relative_freq)

    fig.add_trace(go.Scatter(x=bin_edges[:-1], y=cdf, mode='lines',
                             name="Kumulativna funkcija distribucije",
                             line=dict(color='rgba(200, 50, 50, 0.8)', width=2)),
                  row=2, col=1)

    fig.update_layout(
        height=800,
        width=1000,
        title_text="Analiza prostorne distribucije jakosti električnog polja, E",
```

```
            title_font=dict(size=24),
            showlegend=False
        )

        fig.update_xaxes(title_text=f"Jakost električnog polja [{self.field_strength_unit}]", title_font=dict(size=14), row=1, col=2)
        fig.update_yaxes(title_text="Relativna frekvencija", title_font=dict(size=14), row=1, col=2)

        fig.update_xaxes(title_text=f"Jakost električnog polja [{self.field_strength_unit}]", title_font=dict(size=14), row=2, col=1)
        fig.update_yaxes(title_text="Kumulativna vjerojatnost", title_font=dict(size=14), row=2, col=1)
        fig.update_yaxes(range=[0, 1], row=2, col=1)

        for i in fig['layout']['annotations']:
            i['font'] = dict(size=16, color='white')

        colorbar_trace = go.Scatter(x=[None], y=[None], mode='markers',
                                    marker=dict(
                                        colorscale=[(i/(len(self.color_selector.unique_colors)-1), f'rgb{tuple(color)}')
                                                    for i, color in enumerate(self.color_selector.unique_colors)],
                                        showscale=True,
                                        cmin=self.color_selector.min_value,
                                        cmax=self.color_selector.max_value,
                                        colorbar=dict(title=f'Jakost električnog polja [{self.field_strength_unit}]', thickness=20, len=0.6, y=0.8, yanchor="top")
                                    ),
                                    showlegend=False)
        fig.add_trace(colorbar_trace)

        fig.show()

    def show_pixel_value(self, pos):
        if self.color_selector.color_to_value is not None and self.selected_region is not None:
            x1, y1, x2, y2 = self.selected_region
            scale_x = self.image.shape[1] / self.image_label.width()
            scale_y = self.image.shape[0] / self.image_label.height()
            relative_x = int((pos.x() * scale_x) - x1)
            relative_y = int((pos.y() * scale_y) - y1)

            if 0 <= relative_x < (x2 - x1) and 0 <= relative_y < (y2 - y1):
                color = tuple(self.image[y1 + relative_y, x1 + relative_x])
                value = self.color_selector.get_color_value(color[::-1])
                if value is not None:
                    self.statusBar().showMessage(f"Pozicija: ({relative_x}, {relative_y}), Jakost električnog polja: {value:.2f} {self.field_strength_unit}, Boja: RGB{color[::-1]}")
                else:
                    self.statusBar().showMessage("Piksel izvan definirane skale boja")
            else:
                self.statusBar().showMessage("")

    def zoom_image(self):
        self.zoom_factor = self.zoom_slider.value() / 100.0
        self.display_image()

def main():
    app = QApplication(sys.argv)
    ex = ImageAnalysisGUI()
    ex.show()
    sys.exit(app.exec_())

if __name__ == '__main__':
    main()
```